Preprint
UCRL-JC-149963

Discovery and Classification of Bioinformatics Web Services

D.Rocco, T. Critchlow

September, 2002

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Discovery and Classification of Bioinformatics Web Services

Daniel Rocco
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
rockdj@cc.gatech.edu

Terence Critchlow
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551
critchlow1@llnl.gov

September 2, 2002

Abstract

The transition of the World Wide Web from a paradigm of static Web pages to one of dynamic Web services provides new and exciting opportunities for bioinformatics with respect to data dissemination, transformation, and integration. However, the rapid growth of bioinformatics services, coupled with non-standardized interfaces, diminish the potential that these Web services offer. To face this challenge, we examine the notion of a Web *service class* that defines the functionality provided by a collection of interfaces. These descriptions are an integral part of a larger framework that can be used to discover, classify, and wrap Web services automatically. We discuss how this framework can be used in the context of the proliferation of sites offering BLAST sequence alignment services for specialized data sets.

1 Introduction

The World Wide Web presents a mechanism for unprecedented dissemination and sharing of information among biology researchers. Today, scientists can easily post their research findings on the Web or compare their discoveries with previous work, often spurring innovation and further discovery. The value of accessing data from other institutions and the relative ease of disseminating this data has caused an increase in capacity for collaboration. Increased collaboration produces dramatically larger data sets than were previously available, which require advanced data management techniques for full utilization.

For instance, consider the BLAST [5] family of interfaces, which allow biologists to compare DNA and protein sequences with an existing body of knowledge to find similar sequences in other organisms. Because of its usefulness, most genomics sequence data sources provide a BLAST interface to allow scientists to easily identify homologs of an input sequence. It is practically impossible to determine a priori whether a particular repository will contain homologs of a given sequence; ideally, we would like any search for all known homologs to execute a BLAST query against all sites supporting this type of search.

Unfortunately, there is no common interface or data exchange mechanism for these sites. To perform a search, a scientist must choose a set of sites to query, enter their query into each site, and integrate the results by hand. The problems with this approach are numerous: the scientist may not query the most relevant sites for their search, the search must be entered multiple times, the results of the search must be merged together by hand to obtain an integrated set of results, and if an interface changes or moves, the scientist must figure out where the appropriate interface is and how to query it appropriately.

Our goal is to provide a common interface to the vast and dynamic assortment of bioinformatics data sources. This problem can be divided into three sub-problems: 1) *discovery*, 2) *classification*, and 3) *integration*. Discovery involves identifying new sources of relevant information; classification determines

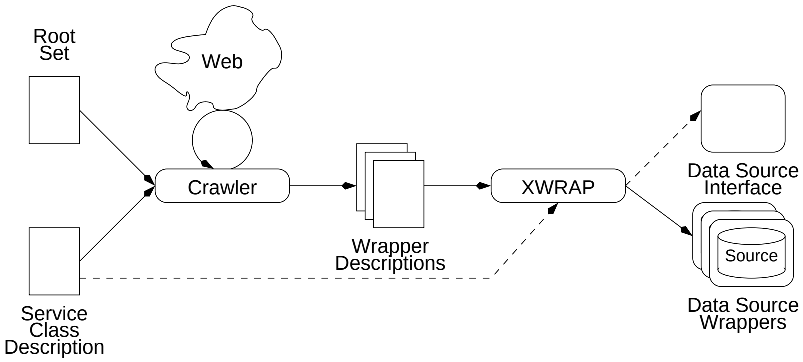

Figure 1: *Source Discovery Overview*

which discovered sources are relevant to a particular query; integration provides a mediation capability to map between sources and produce an intelligently integrated result set.

We propose the use of a *service class description framework* as a solution to these problems. Section 2 outlines our proposed system and discusses Web services and our approach to service discovery. Section 3 introduces the concept of service classes and the role they play in the system. Given a service class description, Section 4 describes how it can be used to generate a wrapper description for a specific site.

2 System Overview

We define a *Web service* as a Web site that provides dynamically generated content, as opposed to "ordinary" static Web pages, which provide visual information using fixed data and offer no support for dynamic interaction. Web services are typically accessed via a client Web browser with HTML forms as the data exchange protocol. While browsers and Web forms are common, many other access methods and implementations are possible.

The bioinformatics community has embraced the Web and Web services wholeheartedly, and many sites [3, 5, 9, 7] offer current research, database information, and message boards for bioinformatics enthusiasts and researchers. In order to utilize these services to their potential, we require automatic tools that will discover new services, classify them into coherent groups, and integrate these groups with a single, seamless point of access. With such a system in place, researchers would enter a query once yet have access to all the latest changes in repositories scattered throughout the world.

Our approach centers on the notion of a *service class*, which is a grouping of Web services based on the functionality they provide. For example, the Web sites Google, Teoma, and Alta Vista could be classified as members of the "Web keyword search engines" service class. Using service classes as an encapsulation tool allows us to reason about an entire class of services; with this approach, we can build applications that rely on the common functionality provided by all members of the class. Differences in implementations of the various class members are hidden from the application and are handled by a class member's wrapper.

Figure 1 shows an overview of our source discovery system. To find Web services of interest, an administrator generates a *service class description*—discussed in the next section—and feeds the description into a Web crawling engine. Starting from a root set of relevant sources, the crawler searches the Web for sites providing a Web service—e.g. those sites that support Web forms. When it finds a site, the Web crawler uses the service class description to analyze the site; if it matches the description, we say that the site is a member of the service class described.

```
<type name="AlignmentSequence" >
  <sequence>
    <element name="AlignmentName" type="string"
             pattern=".{1,100}:" />
    <element name="m" type="SequenceIndex" />
    <element name="Sequence" type="Sequence"
             pattern="[GCAT-]+" />
    <element name="n" type="SequenceIndex" />
  </sequence>
</type>
```

Figure 2: *Sample DNA BLAST type.*

After classifying a site as a service class member, the crawler generates a wrapper description (described in Section 4) that maps the general concepts described in the service class description to the implementation details of the particular site. Once generated, the XWrap [4, 6, 8] wrapper-generation system processes the wrapper description and produces a Java wrapper for the site. The XWrap system also generates a Java Interface definition from the service class description, which allows all members of a particular service class to be queried via a common interface.

3 Service Classes Descriptions

A service class description presents the relevant aspects of the service from the perspective of an external application. The description includes the various data types used by the service, example queries and output, and a control flow graph representing how types can interact. For example, a simplified view of the DNA sequence BLAST service class includes a DNA sequence input type, a DNA BLAST result output type, and descriptions of the intermediate pages. The control flow graph shows the input page being connected to a result page, possibly through a delay page.

Service classes are described using an XML syntax with three main components. The first section in the description is the type definitions, which describe the atomic and complex types needed by members of the service class. The service class type system is modeled after the XML Schema [1] type system and provides the mechanism for declaring the data types that will be needed when processing a site. For example, Figure 2 shows the specification of a DNA BLAST alignment as part of a DNA BLAST service class description. We can view a query line of an alignment as a string of the form

$$I_\alpha\colon\ m_\alpha\, x\, n_\alpha,$$

where I is the sequence identifier, m is the starting sequence number, n is the ending sequence number, x is the sequence fragment, and α is a marker for the current alignment. Thus we might have an example such as:

```
Query:  280 TGGCAGGCGTCCT 292
```

which shows part of a query DNA sequence found in the results of a DNA BLAST.

The second section of a service class description enumerates the navigational paths through members of the service class. Figure 3 visually depicts a candidate control flow graph for a DNA BLAST service. Nodes in the graph represent control points; these are typically pages encountered while interacting with the site, such as the query entry page. The highlighted nodes show entry or exit points and represent a service's interface. Directed edges depict possible execution paths between control points. For this example, entry to

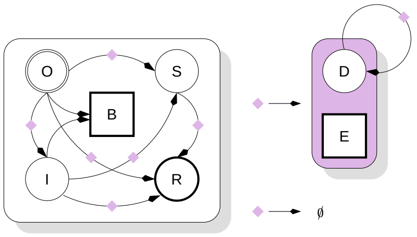

Figure 3: *Control Flow Graph for a DNA BLAST service.*

the service occurs at the double circle “o” (origin), while bold-edge elements are stop points: “r” for result, “b” for DNA BLAST error, and “e” for general error. Other states that might exist in a DNA BLAST service include indirection pages (“i”) used for formatting and tuning results, summary pages (“s”), and delay points (“d”). The shaded inset represents “universal” control points, which can be inserted at diamond-marked edge in the graph. This means, for example, that one or more delay pages might be encountered at any point between the origin state and the summary or results.

The final component of the service class description is the example queries. The crawler uses these examples to test a site and see if it produces reasonable results when queried.

4 From Classes to Wrappers

After classifying and analyzing a site, the system must produce a wrapper that can be used by applications interested in interacting with members of the service class. Such applications expect a single, uniform interface to all members of the class, so every generated wrapper must support the same interface signature despite differences between sites with respect to input parameter names, data values, supported features, process flow, etc. The wrapper must be able to reconcile the application’s expected input parameters with the nuances of the wrapped site and, once the site returns an answer, the wrapper must reverse the process by transforming the result into the expected output format. Finally, the wrapper needs to handle exception conditions gracefully.

As outlined in Figure 1, our system uses service class descriptions as the basis for generating wrapper descriptions. As described in [2], a wrapper description has three basic components: a profile or interface, a model of how the Web service works, and a grounding, which specifies how to access the service. The service class description explicitly provides the information necessary to generate an interface for the wrapper since it includes the type definitions of the inputs and outputs to the system. The crawler’s analysis must use this information to generate the model and the grounding.

The service class definition’s control flow graph provides the basis for the model; however, it is generic and applies to any member of the service class, while the model should be tailored to each site. For example, a particular DNA BLAST interface may not use indirection or delay pages, and thus a wrapper for that interface should not look for them. To map from the control page to the model, the crawler will submit a number of queries to the site and use a heuristic approach to identify corresponding states between the site and the control graph. For example, after identifying an input page for a site, the results of several queries will be tested to determine if they correspond to a BLAST summary page, and indirection page, etc. By probing the site using a variety of queries, different paths in the graph can be explored and the crawler can infer which subset of the graph is employed by the site. With this information, it can produce the model corresponding to the site.

We are presently implementing several components in this system. Currently, work is proceeding along four parallel tracks: refining the service class description language, finalizing the wrapper description format, constructing the crawling and analysis components, and producing an example application.

A complete implementation of this system consists of three subsystems: the source discovery component shown in Figure 1, a user interface, and a data retrieval and mediation system. The user interface is the user's entry point into the system and provides query submission and result display functions. We have described the source discovery component in the previous section. The retrieval and mediation system provides a glue layer between these two systems. It is responsible for managing information about known sources along with the wrappers for those sources. In addition, the mediation component accepts queries from the user interface, passes them to the various wrappers, accepts results from the wrappers, performs result integration, and passes the integrated results back to the user interface. As part of our ongoing research and development work, we are implementing a user interface and mediation layer for the DNA BLAST service class.

5 Conclusions and Future Work

As the web continues to be the preferred means for disseminating scientific information, we are seeing a set of common capabilities, such as homology and keyword searches, being supported by a large number of sites. This observation has been a key motivator for our development of service class descriptions capable of defining the underlying capabilities supported by these collections of interfaces while hiding the implementation differences between specific interfaces. This approach provides an abstract view of bioinformatics sources that allows us to reason about the class rather than being concerned with the intricate details of each interface. Using the concept of service class descriptions, we showed how an integrated DNA BLAST service might operate via a common application to an automatically discovered set of DNA BLAST sources, each wrapped to provide an interface between the application and the data source.

References

[1] XML Schema Part 0: Primer. Technical report, World Wide Web Consortium, http://www.w3.org/TR/xmlschema-0/, 2001.

[2] D. Buttler and T. Critchlow. Using meta-data to automatically wrap bioinformatics sources. In *Objects, XML, and Databases OOPSLA Workshop*, 2001.

[3] DBCAT, The Public Catalog of Databases. http://www.infobiogen.fr/services/dbcat/, 2002.

[4] Georgia Tech. XWRAP Elite Project. *http://www.cc.gatech.edu/projects/disl/XWRAPElite*, May 2000.

[5] W. Gish. BLAST. http://blast.wustl.edu/, 2002.

[6] W. Han, D. Buttler, and C. Pu. Wrapping Web Data into XML. *SIGMOD Record*, June 2001.

[7] International Nucleotide Sequence Database Collaboration. http://www.ncbi.nih.gov/collab/, 2002.

[8] L. Liu, C. Pu, and W. Han. XWrap: An XML-enabled Wrapper Construction System for Web Information Sources. *Proceedings of the International Conference on Data Engineering*, 2000.

[9] National Library of Medicine/National Institutes of Health. National Center for Biotechnology Information. http://www.ncbi.nih.gov/, 2002.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551