Bio-Broker: a tool for integration of biological data sources and data analysis tools


E.T.S.I. Informática, Campus de Teatinos, 29071, Málaga, Spain

SUMMARY
In this work we present an architecture for XML-based mediator systems and a framework for helping systems developers in the construction of mediator-services for the integration of heterogeneous data sources. A unique feature of our architecture is its capability to manage (proprietary) user’s software tools and algorithms, modelled as Extended Value Added Services (EVASs), and integrated in the data flow. The mediator offers a view of the system as a single data source where EVASs are readily available for enhancing query processing. A Web-based graphic interface has been developed to allow dynamic and flexible EVASs inter-connection, thus creating complex distributed bioinformatics machines. The feasibility and usefulness of our ideas has been validated by the development of a mediator system (Bio-Broker) and by a diverse set of applications aimed at combining gene expression data with genomic, sequence-based and structural information, so as to provide a general, transparent and powerful solution that integrates data analysis tools and algorithms. Copyright © 2006 John Wiley & Sons, Ltd.

Received 23 August 2004; Revised 15 November 2005; Accepted 16 November 2005

KEY WORDS: biological mediation; biological dataflow; data analysis tools; XML

1. INTRODUCTION
Technological breakthroughs such as high-throughput sequencing and gene-expression monitoring technology have nurtured the ‘omics’ revolution, enabling the massive production of data. Unfortunately, this valuable information is often dumped in proprietary data models and specific
services are developed for data access and analysis, without forethought to the potential external exploitation and integration of such data.

However, dealing with the exponential growing rates of biological data is a simple problem when compared with the problem posed by diversity, heterogeneity and dispersion of data [1]. Nowadays, the accumulated biological knowledge needed to produce a more complete view of any biological process is disseminated around the world in the form of biology sequences and structure databases, frequently as flat files, as well as image/scheme-based libraries, Web-based information with particular and specific query systems, etc. Under these conditions, this plethora of interrelated information becomes difficult to use, pointing to the integration of these information sources for unified access, independently of possible internal changes, as a clear and important technological priority.

The database community has been engaged for many years with the problem of integrating heterogeneous data sources [2]. A first approach was based on wrappers, which act as interfaces for the data sources, translating their structure into a common model [3,4]. Later, the focus moved to the use of mediators for encapsulating the knowledge needed for evaluating a query over multiple wrappers. The wrapper–mediator approach sets up an interface for a group of (semi-structured) data sources, amalgamating their local schemas into a global schema and integrating the information of the local sources. As a result, the views of the data that mediators offer are coherent, producing semantic reconciliation of the common data model representations carried out by the wrappers. Some examples of the wrapper–mediator systems are Amos [5], Tsimmis [6], Disco [7] and Garlic [8]. In the specific field of biological data there are the following examples: TAMBIS [9], BioDataServer [10], KIND [11], BioZoom [12], BioKleisli [13] and DiscoveryLink [14]. With the advent of XML, some of the mediator systems have evolved toward the XML standard (Amos and Tsimmis), whilst other projects were initially XML-based, such as MIX [15] and MOCHA [16].

We have designed an architecture called Bio-Broker for XML-based mediator systems and developed a framework for assisting systems developers in the construction of mediator services that integrate heterogeneous data sources. Our architecture allows the easy development of Extended Value Added Services‡ (EVASs). This is especially suitable for environments where users have to share tools with which they are unfamiliar. The final user will have a view of the system as a single data source and furthermore, with the capability of easily selecting, adding and modifying EVASs for enhancing of query processing.

In the bioinformatics context, researchers often use a variety of tools for data analysis. Each of these tools have their own interfaces, data formats and semantics, and probably need to be interconnected in a workflow in order to define a processing pipeline, thus becoming complex virtual bioinformatic machines [17]. Today, users build these workflows by hand, which may be a source of errors and involves a time cost. It is important to automate this process to free biologists of this repetitive task. We propose the use of EVASs for making the workflow construction easier. Each EVAS encapsulates a software tool as a black box that may be used as a building block, which can be connected to another EVAS, building the desired workflow in an almost automatic way. EVAS may be interconnected without requiring the knowledge of the internal details related to the encapsulated tools. Furthermore, it allows private software tools to be shared in a ‘pay per use’ way of operation.

‡An EVAS is an external software tool integrated in the mediator (i.e. a data mining algorithm).
The flexible use of EVAS is an outstanding characteristic of our system, which is not provided by traditional or biological mediators such as TAMBIS, BioDataServer, KIND, BioZoom or BioKleisly.

Among closely related developments to Bio-Broker, we find BioMOBY, which provides a way to create Web services as well as a central registry that has information on all the services that are available (BioMOBY). In fact, the main difference between BioMOBY and Bio-Broker lies in that Bio-Broker is a mediator system, i.e. an intermediary between data sources, services and users that offers a view of the whole system as a single resource, where EVASs are readily available for enhancing query processing (see complementary material at http://uranos.khaos.uma.es/mediator for details). Another related project is myGRID, which also provides a prototype system named Taverna for building workflows (Taverna).

However, Taverna only offers a way to build and execute workflows, but it does not include integration capabilities. Thus, BioBroker combines traditional data integration through an integration schema with workflow definition and execution (see complementary material for details).

As an example of such integration procedures we have developed a biological instance of the framework in the context of gene-expression data. In this context, the query space is, in practical terms, unbounded and dynamic. So the problem is not only to integrate diverse and scattered data sources but also to find a way to analyse the queries, to identify with which service or database should be solved, as well as to integrate dispersed services and apply them to the selected set of data.

This biological mediator (Bio-Broker) integrates genetic information and various processing mechanisms developed by different users and groups through a graphical interface that allows fast and intuitive ‘wiring’ of EVASs components, expanding the functionality and enabling the easy incorporation of new procedures to customize the system for specific concerns. Bio-Broker allows uniform access to data stored in different databases: presently these include SWISS-PROT [18], EMBL [19], MICADO [20], PDB [21], DIP [22] and BIND [23] as well as to several processing services such as BLAST [24], AnaGram [25], Frags [26] and AsocRules [26]. The end result of this service is the ability to combine gene-expression data with genomic, sequence-based and structural information, so as to supply a general, transparent and powerful solution. As a proof of concept, Bio-Broker has been also used to extend the capacity of engene, a versatile, Web-based and platform-independent exploratory data analysis tool for gene expression data (see [27] for more details).

Here we describe both the architecture design and the development mechanisms needed to build up the mediator. We also define the problems selected to illustrate the system. We then describe a mediator system, Bio-Broker, which offers enhanced query capabilities beyond gene-expression clustering.

2. SYSTEM AND METHODS

2.1. An XML based architecture for mediators design

The mediator system architecture [28,29] is based on the use of XML standards. The key characteristics of the architecture and its main components are as follows (see Figure 1).

(1) The architecture is XML-based. XML [30] is itself a standard format for the representation, interchange and validation of data and, together with the XQuery query language [31] and the XML Schema [32], forms a complete data model for dealing with information on the Web.
XML Schema [32] and RDF Schema [33] are used for describing all the metadata necessary for query processing in this architecture as well as for expressing the data resulting from the queries.

(2) XQuery is the query language used by the mediators. All the clients connected to a mediator must send a valid XQuery query to the mediator and will receive an XML document. The client can be any sophisticated query generation mechanism and the only requirement is that it must send a valid XQuery query independent of how this query is generated. A generic tool is available for speeding up the development of specific graphical query interfaces.

(3) The query processor is the component that receives a query and generates a sub-query for each of the data sources involved in the query. It analyses the data requirements of the query, identifying the data sources and generating a global plan for evaluating the query. The global plan contains
information on the various alternatives for solving the same data requirement. Later, the global plan is fragmented into one sub-query for each of the selected data sources.

(4) Metadata are used to express all information needed for the above process. This includes information both on the logical schemas produced by the wrappers that encapsulate the data sources, and on the semantic equivalence of the terms produced by the data sources (the integration schema). We also store information about the query processing capabilities of these wrappers and about the location and availability of the data sources for optimizing access to the data. Specific user information such as privileges, EVASs usage and preferences is also stored.

(5) EVASs are all those processes that allow the user to further filter or post-process the queries. In a sense EVASs are modelling arbitrarily complex repetitive processes appearing on the data obtained from the data sources and shared by several users. Obviously these processes (e.g. BLAST, AnaGram, Frags) being application specific are not efficiently incorporated into the query language. On the other hand, being repetitive tasks performed by several users, their automation and optimization should be assisted by the mediator.

(6) Wrappers are the components that translate the various data sources being integrated into the common data model (XML/Xquery). Wrappers hide the internal complexity of the data sources and offer, in a controlled manner, their query capabilities. There is one wrapper for each data source integrated into the mediator. Wrappers receive XQuery queries and produce XML documents that conform to the XML schema (integration schema).

2.2. From architecture to mediators

A framework has been developed for simplifying the mediator construction task. The mediator instantiation process involves the construction of a particular application of the framework by the development of wrappers and the use of metadata for the configuration of the different mediator components. Wrappers are the means of translation between the reference model of the architecture (XML) and the data source model. Once the mediator has been constructed, the mediator user can dynamically manage the EVASs adding or removing them as necessary.

The main tasks involved in mediator configuration are as follows.

Wrapper construction. Specific wrappers are necessary for each data source integrated into the mediator. To simplify this task, our system offers ‘wrapper construction schemes’ for three of the more frequent types of data sources (relational databases, XML documents and HTML documents).

Mediator configuration through metadata. Metadata will be used for adapting the mediator components. Specifically they will be used to configure both the location and integration schemes.

• The location scheme links the wrappers to the mediator. It is used for identifying the data source of a query and selecting the appropriate wrapper.
• The integration scheme specifies the data translations between the data model used by the client and that offered by the wrappers. So, for each database field, it specifies by Xpath its location into the XML document given by the wrapper. Furthermore, it gives information on the data source where such is available.
Figure 2. Developing and publishing of E-BlastP, an EVAS which performs a BlastP-search using a query protein sequence and some (user selected) parameters. E-BlastP receives the input as an XML document. Therefore, before using BlastP, E-BlastP parses and translates this input. BlastP results are then translated into an XML output document. The right-hand side of the figure provides a snapshot of the graphical interface for building workflows by pipelining EVAS.

2.3. Dynamic integration of EVASs

An EVAS is a software tool adapted for being connected to the mediator. EVASs are not in the mediator core. It is not possible to know in advance how to build the EVASs structure. EVAS are linked dynamically during the use of the mediator system. Therefore, although the EVASs configuration is made using metadata, the framework offers to the mediator-users a management tool to connect, arrange and remove EVASs without any information on the mediator internal architecture, thus, enabling a wide range of processing alternatives to be specified for relating—in different ways—diverse sources of biological data.

The integration of EVAS with the mediator consists of two main phases.

1. Creation of the EVAS. The tool (program) to be integrated into the mediator has to be adapted for obtaining a new program (EVAS) with three parameters, an input XML document, an XQuery query and a list of zero or more parameters (those that correspond to the tool). The resulting EVAS will use these three parameters for calling the original tool, thus obtaining an output XML document. The input and output XML documents must conform to the schemas, which determine its connection in the workflow (see Figure 2).

2. Connection of EVAS to the mediator. Once an EVAS has been created, it has to be linked to the mediator. This implies the following tasks.

(a) Publication of the EVAS as a Web service. It will be made as a Web method namely Query, or as a server side program (CGI, ASP, JSP, etc.)
(b) Registration of the Web service in the system, sending information about this service to the mediator administrator.

(c) Obtaining a DLL library, which performs a call to the Web service. This library allows the user to add services dynamically to the mediator without changes in its source code. The library will be automatically generated by the mediator administrator, using previously implemented templates.

(d) Registration of the library in the system. For this, some metadata, including name of the library, name of the service, and input and output XML schema will be added.

When a set of EVAS has been developed and connected (see Figure 2) to the system, users can create workflows using a graphical tool provided by the system. This tool has been developed using C# (the Microsoft programming language) and makes use of an internal representation for workflows. This tool allows users to connect databases, EVAS and the data integrator. Once the mediator has been developed and configured, it is ready for use by the final clients. The clients can use the mediator either through a programming interface or through a graphical interface (for details about this tool see the supplementary material).

This architecture does not impose any restriction on the definition of EVASs or on the way in which they are connected in workflows. Generic EVASs can be defined and they can also be replicated and/or specialized to facilitate availability and/or communication efficiency. For example, running a BLAST query on a custom set of genes would require transferring the entire set of genes for comparison against the BLAST server (typically many gigabytes). In this case, it would be much more efficient to link this EVAS to a database that already has the necessary genes. In this case, a simple service can recover the genes from the database avoiding communication overheads.

2.4. Biological database integration: Bio-Broker

We present a sample use of the mediator system for the integration of gene expression applications. In general, clustering is one of the most common forms of analysis for such applications, and once the data has been clustered, additional post processing analysis has to be performed to extract relevant knowledge.

In a sense, clustering analysis is incomplete without integrating it with functional information contained in nucleotide and protein databases (both at sequence and structure level) as well as with information contained in emerging pathways and assembly databases. A common option is the static integration of these very diverse data sources (equivalent to a pre-annotation process). However, what is really necessary and much more interesting is the dynamic integration of services onto this data so as to obtain valuable information from the diverse sources.

At present, Bio-Broker integrates many different types of data sources: nucleotide (EMBL) and protein (SWISS-PROT) sequence information [18]; the worldwide repository of three-dimensional structures of biological macromolecules—the Protein Data Bank, (PDB) [34]; the MICrobial Advanced Database Organization (MICADO)—a relational database dedicated to microbial genomes [20] and functional analysis of Bacillus subtilis [35]; the Database of Interacting Proteins (DIPTM); and the Biomolecular Interaction Network Database (BIND) designed to store full descriptions of interactions, molecular complexes and pathways.
These databases have been selected on the basis of differences in content, format, access mechanism and geographical location. Our intention is to show how these very diverse data sources can be easily integrated using the proposed architecture and how the possibility of easily adding EVASs allows the user to mix different tools for obtaining enhanced data processing capabilities.

Services requested of the mediator help to recover information associated with a given set of clustered genes, independent of the way in which these clusters were obtained, on the basis of their expression levels by association rule discovering or by whatever other appropriated method.

3. RESULTS

3.1. Bio-Broker architecture

Figure 3 shows the complete architecture of Bio-Broker. A user interface (optionally a graphical one) is used for accessing the services. The queries, expressed in terms of the integration schema, are sent in XQuery to the server, which divides it into different subqueries. The subqueries are sent to the different databases. Each database has its own query and XML result document construction mechanism (i.e. database-specific wrappers). The service receives the sub-query results. At this point, it is possible to apply an EVAS to process the sub-query results. For example, a new EVAS can be applied to the first EVAS process result, and so on.

The results of each sub-query (with or without applied EVAS processing) are integrated, removing duplicates and inconsistencies. Duplications can be removed by taking advantage of the mappings between the integration schema and the resource schemas. Data that the mediator receives is expressed in terms of the mediator’s integration schema and the mappings identify any duplicated items and remove one of them from the integrated data. Note that EVAS processing does not include duplicate removing, this step is performed in the integration step.

Next, the integrated solution can be post-processed by a set of EVAS. The final result is then sent to the user interface. It is worth observing that where and how these EVASs are implemented is completely irrelevant: they are just Web services integrated into the mediator system. In the architecture these modules are used as a ‘black box’. They can even be developed by other people. To include a new EVAS one only needs to know its address (IP) and the service must conform to the DTD/schema.

The integration schema of Bio-Broker stores both the data provenance (the database(s) from which the data can be obtained) and the query capabilities (the elements of the resource scheme by which users can make queries and the expressiveness of the query language). Data provenance is an important issue due to the necessity of assessing the quality of data in the biological context. The quality of data obtained from proprietary Web services, not being up-to-date or well-ordered would not be as good as that obtained from well known, widely used biological Web services, such as EMBL, PDB, etc. Therefore, data provenance is not only stored in the integration schema but also annotated in the answer so that the source can be identified, and the users can infer its quality according to their confidence in the data source. Thus, we can for example annotate the GeneName field with metadata showing that it can be obtain from EMBL, GenBank or MICADO databases (see the supplementary material).

Please note that we have not defined an architecture with multiple middle-layers; we have rather defined an architecture with three layers: user interfaces, the mediator and wrappers. It is also important to note that although EVASs are fully integrated in the architecture they are not mandatory and so do
Figure 3. Bio-Broker architecture: data access, pre-processing, integration, post-processing and user interfaces.
Figure 4. Instance of our integration architecture with three EVASs.
not represent another architectural layer; therefore, it is not necessary to include a set of EVASs for each query processing task. Figure 4 shows an instance of our architecture for three EVAS, applied after the integration step. Addition of EVASs in the integration process does not impose a performance penalty on the query processing step. Performance is simply the sum of the time of making use of these tools and the time required to retrieve the data. Furthermore, the end-user cost of using these tools is always less than using them manually, since the results are integrated. That is, the repetitive manual application of filters and transformation tools to data retrieved from data sources has a high cost; this is decreased by the mediator, which applies them automatically. Besides, execution of workflows is performed automatically, taking advantage of metadata, so the cost of maintenance is limited to definition of workflow according to user needs.

3.2. Bio-Broker services

As a demonstrable ‘proof of concept’ that can be evaluated and tested in the field we present the integration of gene expression data with the biological databases described in Section 2. The data set described by [36] will be used. This data set is composed of the cDNA expression levels of 4005 Bacillus subtilis two-component regulatory systems in which the overproduction of a response regulator of the two-component systems, coinciding with a deficiency of its cognate sensor kinase, affects the regulation of genes, including its target genes. The genome-wide effect on gene expression caused by the overproduction was analysed on 24 two-component systems (http://www.genome.ad.jp/kegg/expression).

The engene platform (at http://chirimoyo.ac.uma.es/bitlab) was used to cluster this subset of gene expression data. The mediator system is hosted at IP address http://uranos.khaos.uma.es and is available by using the Web services (see http://uranos.khaos.uma.es/mediator). The Web method ‘connect’ supplies an identifier to dialog with the system. The WSDL description of the method for querying has two available methods ‘ConsultCompress’ and ‘Consult’. The latter produces a ‘string’ XML document, while the former compresses the data, and the client must use Xmill to decompress. Parameters for both methods are an array with the query (queries), the number of queries, the userid obtained when connected to the system, a Boolean array that specifies the services to use, and the parameters needed for each service (see the complementary material).

Table I describes four samples of implemented queries. These examples describe different levels of complexity, and the different databases and services that can be requested and combined in a simple query. They have been designed and published as services in http://uranos.khaos.uma.es/mediator, thus we provide biologists with an easy Web interface to solve common queries that they often need to solve.

The first service (Service A in Table I) sends a request to the mediator system for information about the ‘genomic context’ of clustered genes, whose more straightforward definition is based on the physical proximity of genes in the chromosome [37,38]. It is well accepted that neighbouring genes in bacteria are often functionally related, so drawing the gene/cluster distribution provides additional information for interpreting expression data. This service will work on MICADO, which contains detailed information on Bacillus subtilis. This is a very simple request and in a way is equivalent to a pre-annotation process that can be used to include information for analysis from very different sources in the data collection (see Figure 5). Additional information about the query formulation can be found in the complementary material.

Table I. Requested services. Four examples are shown to describe different levels of complexity, and the different databases and services that can be requested and combined in a simple query. The following labels have been used for metadata information: \(<S>\) Species, \(<G>\) Gene cluster, \(<P>\) Prefix length, (BS = Bacillus subtilis).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Gene/Cluster distribution throughout the genome. Clusters were obtained from gene-exp. data. Chromosome zones involved in specific biological processes will be highlighted</td>
<td>Given a set of DNA sequences (same organism), related by their expression data, to end up with another set of over represented words identified at the up-stream positions of the sequences.</td>
<td>Given a set of related DNA sequences (from service [B]) to end up with another set of related sequences and its associated KW from which to identify association rules.</td>
<td>Given a collection of consensus patterns from a set of protein sequences (with their PDB codes) to end up with a structural motif.</td>
</tr>
<tr>
<td>Input</td>
<td>S, G</td>
<td>S, G, P</td>
<td>S, G</td>
<td>G, PDB1d, start, end (from service [C])</td>
</tr>
<tr>
<td>Example</td>
<td>(&lt;S&gt;) BS (&lt;G&gt;) atpA atpB (&lt;G&gt;) (&lt;S&gt;)</td>
<td>(&lt;S&gt;)BS (&lt;G&gt;) atpA atpB (&lt;G&gt;) (&lt;S&gt;) (&lt;P&gt;) 500(&lt;P&gt;)</td>
<td>(&lt;S&gt;)BS (&lt;G&gt;) atpA atpB (&lt;G&gt;) (&lt;S&gt;)</td>
<td>(&lt;PDB1d='1com' start=46 end=56/&gt;</td>
</tr>
<tr>
<td>Database</td>
<td>MICADO</td>
<td>EMBL</td>
<td>EMBL, SWISSPROT</td>
<td>PDB</td>
</tr>
<tr>
<td>Procedure</td>
<td>Send a query to MICADO database</td>
<td>Obtain entries from EMBL, then apply an EVAS to obtain up-stream sequence. Then apply Prefix and Anagram.</td>
<td>Send a call to EMBL, then apply BlastP to each entry to find related sequences. Return that sequences an its associated keywords and PDB codes- when available</td>
<td>Send a call to PDB to obtain the PDB entry. Then apply a procedure to obtain positions and atoms.</td>
</tr>
<tr>
<td>Method</td>
<td>ODBC call</td>
<td>CGI call</td>
<td>CGI call</td>
<td>CGI call</td>
</tr>
<tr>
<td>EVAS</td>
<td>ChromoDist</td>
<td>Prefix, Anagram</td>
<td>BlastP, Frags, AsRules</td>
<td>Cut3D, 3Dalign</td>
</tr>
</tbody>
</table>
### Table I. Continued.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EVAS</strong></td>
<td>Displays a chart with the gene distribution on the chromosome.</td>
<td>Prefix: given an EMBL entry in XML, to end up with a the up-stream sequence.</td>
<td>BlastP: given an EMBL entry, search by similar protein sequences in SWISS-PROT, Return the sequence and its associated keywords.</td>
<td>Cut3D: Given a PDB entry and a range of residue positions, return the atom (or aminoacid) 3D coordinates corresponding to these residues. 3Dalign: Align two 3D structures and compute the rmsd distance between them.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Anagram; Compute the distribution of over-represented words of length K.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frags; is a standalone program whose output consist of statistically significant fragments from the input sequence (roughly named consensus matches).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AssRules: Association rules discovering from a set of patterns and KWS identified from a given set of sequences.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 5. Graphical representation of output from Service A. Red and green colours have been used to represent up- and down-regulated genes, and forward and backward gene direction is represented as out- and in-side the chromosome. Position is counted clockwise. A clear proximity relatedness can be observed with potentially useful biological information.

The objective of the Service B is for given a set of related DNA sequences, clustered by their gene expression data, to find another set of over-represented K-fragments identified from the up-stream positions that could represent putative promoters or activation signals of those genes. In this case, there are two very simple EVASs added to the pipe: Prefix and Repeats, the former to obtain the up-stream fragments from the DNA sequences, and the latter, a tailored algorithm, to identify over-represented strings of length K from those up-stream sequences.

Service C is more complex, oriented to obtaining additional information not only about the original sequences, but about sequences related to the original sequences. In this case the input is also formed by a collection of DNA sequence identifiers and the final output is a collection of Keywords to be used in an association-rules-discovering procedure. The necessary steps for this exercise are as follows. First, from EMBL retrieve the accession numbers of the protein sequences that correspond to the collection of genes (e.g. /db_xref="SWISS-PROT:P37800"). Second, obtain the sequences and keywords corresponding to those proteins. Third, BlastP is launched to complete a database search for similar sequences using each of the original proteins as query sequences. Finally, obtain sequences, keywords and identifiers of the similar sequences. These are used as input to the Frags procedure, which is able to detect statistically significant patterns from the collection of sequences. Sequences, patterns and keywords are used as input to ASSRUL, a procedure to detect association rules (available in the Engene platform) that correlate the presence of some patterns with functional keywords. This is a complete exercise that demonstrates the ability to use EVASs in the pipeline. This allows the user to dynamically incorporate additional information into the gene expression data.

Finally, in Service D, we explore the possibility of combining gene-expression data with structural information. In this case we use the fragments obtained as partial output in Service C, and seek to verify
if these fragments—with strong association at the sequence level—are also associated at structural level. To this end, the service receives as input a collection of PDB codes together with the start and end positions of the fragments and ends up with three-dimensional (3D) alignments of such fragments.

As test case for Service C we have used the YesM-YesN two-component regulatory system, composed of 21 up-regulated genes and three down-regulated genes as reported by [36].

We will now describe with some more detail Service C, which has a similar procedure to Service B. The objective is to obtain additional functional information both from the original sequences and from those sequences related to the original sequences. In Table II we present the query and results from the mediator. First, the mediator requests the protein sequences and its associated keywords corresponding to a given collection of genes. Results from this request are used to launch a Blast database search (see Supplementary material for detailed parameters).

The Blast procedure retrieves a set of related sequences, and the mediator is used to retrieve the sequence and its associated keywords (see Table III, and additional information in the complementary material). Using these sequences, a data mining procedure (Frags) is used to detect statistically significant patterns from the collection of sequences Finally, the ASSRUL procedure is used to detect association rules (available in the engene platform).

From the implementation point of view, the Service C construction requires the creation of a wrapper for EMBL, and the creation and configuration of three EVAS (E-BlastP, E-Frags and E-ASSRUL) around the three mentioned tools: BlastP, Frags and ASSRUL (see Figure 6). These EVAS are published as Web services (WSE-Blastp, WSE-Frags and WSE-ASSRULL), finally they are interconnected in Bio-Broker using a graphical tool provided by the framework, and the desired workflow is obtained.
Table II. XML string output for Service C. This service request for the protein sequences and associated functional keywords of the genes belonging to a collection of clusters (non-informative keywords such as ‘Hypothetical protein’ and ‘Complete proteome’ have been removed). For some cases Blast output clearly shows functional relationships between the query sequence (in this case the protein coded by yesS gene) and a set of by similarity related sequences. In other cases, the keywords are less informative (e.g. compete proteome) and no-functional information is obtained by similarity. In these cases, the sequences are used again in a pipeline workflow as input to the Frags procedure in which short stretches of sequences resembling putative patterns are detected.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<result>
  <query>
    FOR $Iteration IN INTEGRACION WHERE
    ($Iteration/Organism/data()='Bacillus subtilis' AND
    ($Iteration/GeneName/data()='lplB' OR $Iteration/GeneName/data()='licC' OR
    $Iteration/GeneName/data()='opuBD'))
  </query>
</result>
```

DOI: 10.1002/spe
Table III. The rules’ significance can be established upon a variety of parameters calculated by the system: confidence (probability of rule satisfaction), support (examples in data that are covered by the rule), coverage (examples in data that are covered by the antecedent of the rule), improvement (how much more frequent is the occurrence of the rule than normal) and leverage (additional examples covered by the rule above those expected).

<table>
<thead>
<tr>
<th>CONF</th>
<th>SUPPORT</th>
<th>COVERAGE</th>
<th>IMPROV</th>
<th>LEVERAGE</th>
<th>ANTECEDENT</th>
<th>CONSEQUENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.8018</td>
<td>1.8018</td>
<td>14.8</td>
<td>1.6801</td>
<td>[+PAT64]</td>
<td>G-PROTEIN, COUPLED RECEPTOR, GLYCOPROTEIN, MULTIGENE_FAMILY, TRANSMEMBRANE</td>
</tr>
<tr>
<td>100</td>
<td>1.3514</td>
<td>1.3514</td>
<td>6.2535</td>
<td>1.1353</td>
<td>[+PAT40]</td>
<td>GLYCOPROTEIN, HYDROLASE, MEMBRANE, NERVE, NEUROTRANSMITTER, DEGRADATION, SERINE_ESTERASE, SIGNAL, SYNAPSE</td>
</tr>
<tr>
<td>100</td>
<td>1.3514</td>
<td>1.3514</td>
<td>24.667</td>
<td>1.2966</td>
<td>[+PATS]</td>
<td>4FE-4S, IRON-SULFUR, LYASE, MITOCHONDRION, TRANSIT_PEADE, TRICARBOXYLIC_ACID_CYCLE</td>
</tr>
<tr>
<td>100</td>
<td>1.5766</td>
<td>1.5766</td>
<td>6.6269</td>
<td>1.3387</td>
<td>[+PAT43]</td>
<td>CALCIUM, CARBOHYDRATE_METABOLISM, GLYCOSIDASE, HYDROLASE, SIGNAL</td>
</tr>
</tbody>
</table>

The possibility of easily connecting EVASs allows one to solve these examples without requiring knowledge of the internal details of the tools connected to the service as EVASs. This characteristic is especially suitable for allowing different users to share tools with which they are unfamiliar.

Finally, as a test case for Service D we have structurally compared two protein structures: 1dhr and 4tgf (the system has no more limitation than computational resources to deal with larger sets of structures). As commented previously, the set of PDB-codes can be provided directly by the user or be a partial output from Service C. As can be observed, in Figure 7 the core of the fragment resides in the two anti-parallel beta-strands, but additional residues from very different (and partial) two-dimensional structures are also aligned. In this case it represents a random geometrical relationship.

4. DISCUSSION

In this work we have presented an architectural design that allows easy and transparent integration of several heterogeneous sources of information. This architecture is especially suitable for collaborative environments where users wish to include their own specific software tools (EVASs) with minimal intervention and expertise required to manage the system. The inclusion of EVASs in the data flow is one of the features of the architecture and allows filtering, restructuring and processing data without requiring any knowledge of the EVASs implementation.
Figure 7. A 28 residues length 3D-local alignment between 1DHR, pdb code for Dihydropteridine Reductase structure (236 residues) and 4 Transforming Grow Factor (4tgf) (50 residues) obtained using our 3D align tool (a geometrical hashing based tool for aligning 3D structures). The root mean squared of the distance between alpha-carbon positions is 1.58. On the left, there is a ribbons model for the whole unrelated structure, and on the right the local fragment.

Our framework provides a base set of templates that can be readily adapted and refined to meet user needs, incorporating new databases and algorithms for (bioinformatics) integration.

The framework has a modular organization with flexibility for selecting and combining specific EVASs for specific needs. These EVASs designs are reusable for repeated tasks and can be shared with other users.

In this architecture, wrappers are created manually and added to the mediator modifying its source code. We are addressing our efforts to a dynamic mediator through semantic mediation. It allows one to use semantic information about data sources, such as query capabilities, data provenance, data’s schema, etc. The main goal is to provide a method for adding wrappers without source code modifications. A secondary goal tries to provide a tool for automatic wrapper generation. There are a lot of works that treat this problems, but we think that they are, in the most cases, incomplete tools or they produce monolithic wrappers that can not be reused (either as a complete or a partial interface) in other mediators.

Different use cases have also been developed to demonstrate how this architecture and its associated framework allow the rapid development of domain specific applications. The development of Bio-Broker is a proof of concept of the suitability of this kind of architecture for the bioinformatics domain. In particular, the usefulness of the mediator system is demonstrated by a diverse set of applications aimed at combining expression data with genomic, sequence-based and structural information, so as to provide a general, transparent and powerful solution that goes beyond traditional gene expression data clustering.
We are currently working on the integration of gene expression data with pathways information, which would be a significant development. There are several metabolic databases available that contain several hundred manually drawn pathway maps, but in some cases their static and separate diagrams do not provide sufficient flexibility to integrate the information.

ACKNOWLEDGEMENTS

This work has been partially supported by grant QLK3-2001-01473 under the EU sub-programme area ‘Quality of Life and Management of Living Resources’—Key Action ‘The Cell factory’ and the grant TIC2002-04586-C04-04 of the Spanish Ministerio de Ciencia y Tecnología. The authors would like to thank Dr. Andrés Rodríguez and Dr. Antonio Pérez for carefully reading and recommendations.

We thank the reviewers for their comments and help towards the improvement of the paper.

REFERENCES


