COBITIS

COMPUTATIONAL BIOLOGY TOOLS INTEROPERABILITY SCHEME

Uberto Pozzoli[1]
Matteo Cavalleri[2]
Giorgia Menozzi[1]
Manuela Sironi[1]

Introduction

The number of computational algorithms in biology, their availability and their utilization is steadily increasing. Nonetheless to use them from software applications and computational systems still requires programming and data format conversion. Moreover their accessibility from web interfaces, while expanding their diffusion, makes even harder their programmatic use. On the other hand the complexity of the tasks they are involved in, almost always, requires integration between different methods and repetitive runs on different datasets.

[1] Bioinformatics Lab - Scientific Institute IRCCS E. Medea – Bosisio Parini (LC) – Italy
[2] Bioengineering Lab - Scientific Institute IRCCS E. Medea – Bosisio Parini (LC) – Italy

To face these problems, primarily in our lab, we are developing a set tools based on XML and SOAP. Our goal is to obtain a system able to meet the following requirements.

- Easy programmatic access.
- Easy data interchange and deployment.
- Easy web interface implementation.
- Optimization of networked hardware resources.
- Programming language and platform independence.
- Utilization of free/open source software.

Given the focus on programmatic access, the definition of ontologies on data types is not one of our target.

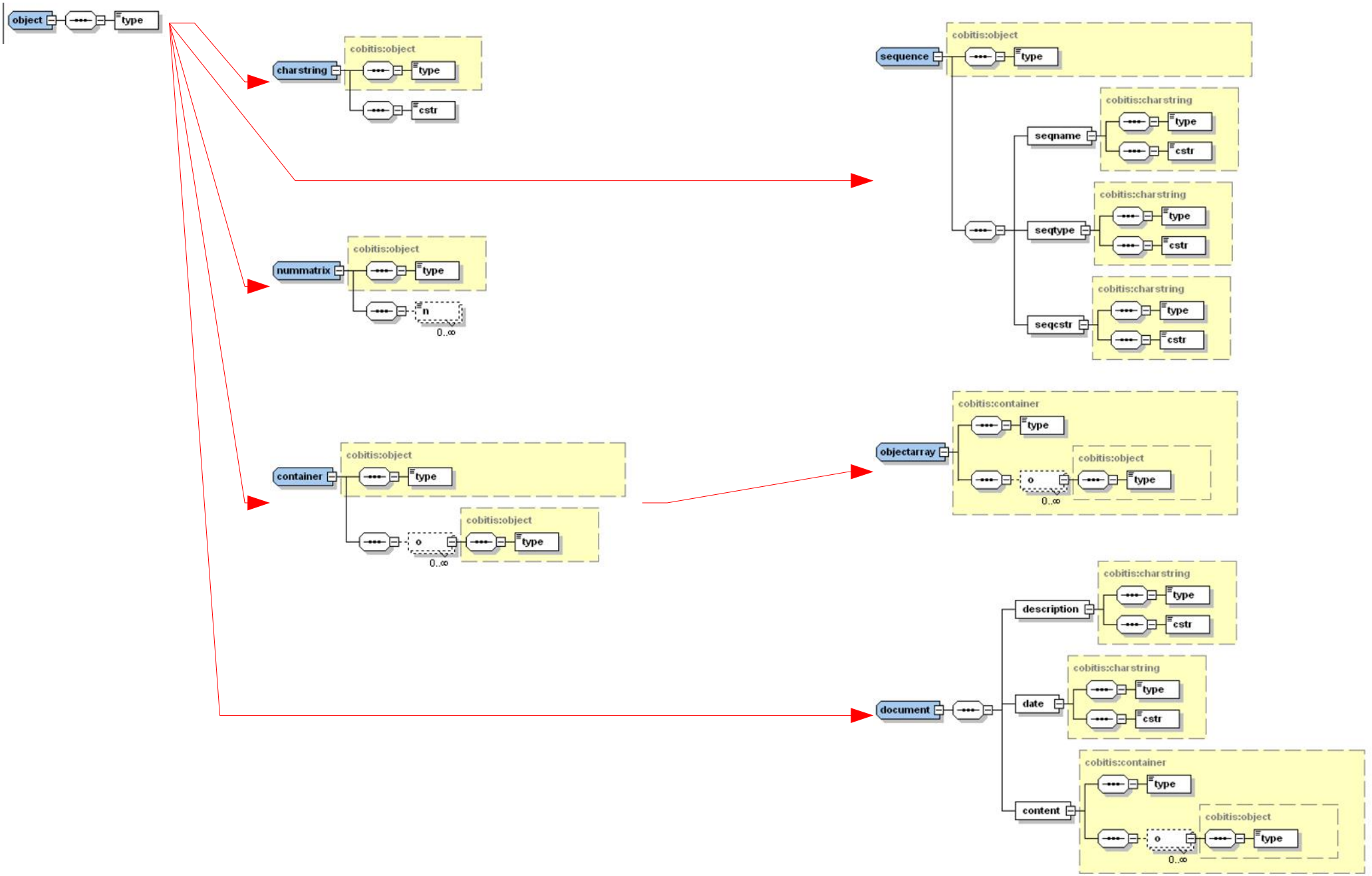
The core of Cobitis is the XML scheme reported in the diagram on the left.
It defines a hierarchy of simple data types that ideally can describe any data type and structure. The most important feature is the definition of the CobiContainer element that allows to virtually represent any data structure.

An ANSI C++ hierarchy of classes maps each data type defined in cobitis, allowing data manipulation, XML and SOAP encoding/decoding. These classes constitute the code base on which the CobiServer and CobiClient modules relies but they can also be used in algorithm implementation.

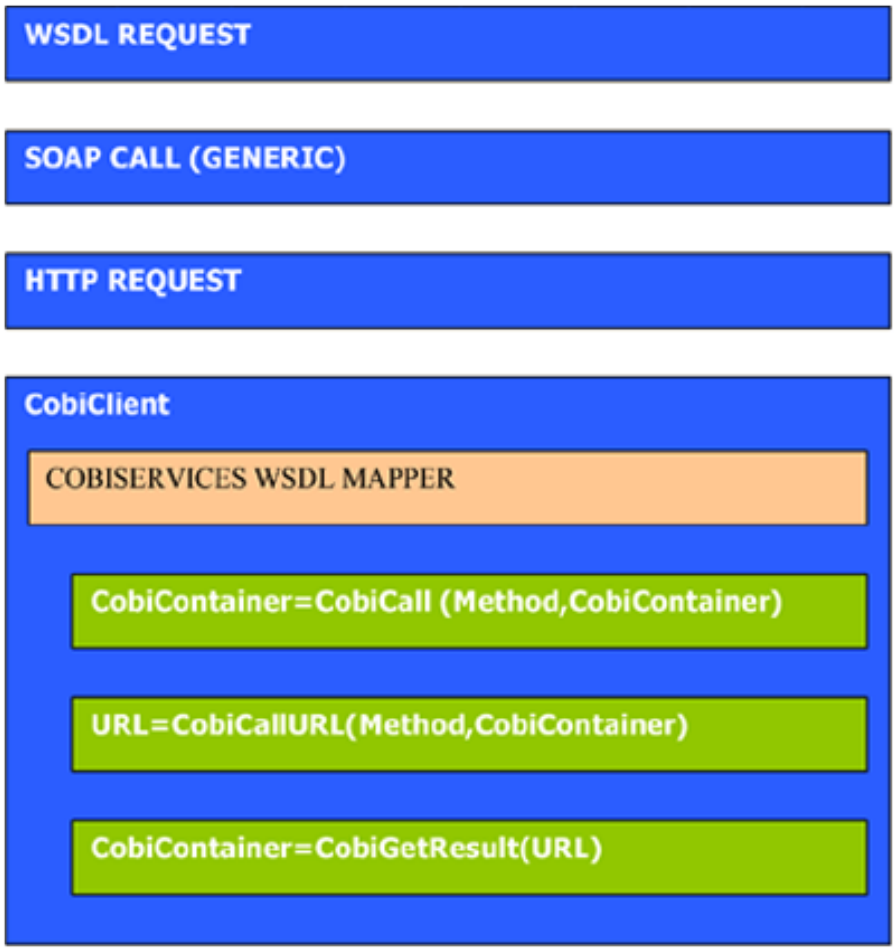
It is straightforward to implement layers between Cobitis Classes and data structures defined elsewhere (i.e. NCBI toolbox, R, Matlab etc); in this way cobitis based application can easily access algorithms already implemented.
On the other hand it is also possible to access methods based on Cobitis from other systems (i.e. R, Matlab, BioPerl).

The core module is based upon STL and gSOAP, a freely available, open source package. Particular care have been given to minimize and isolate code dependency.

COBITIS – XML Scheme

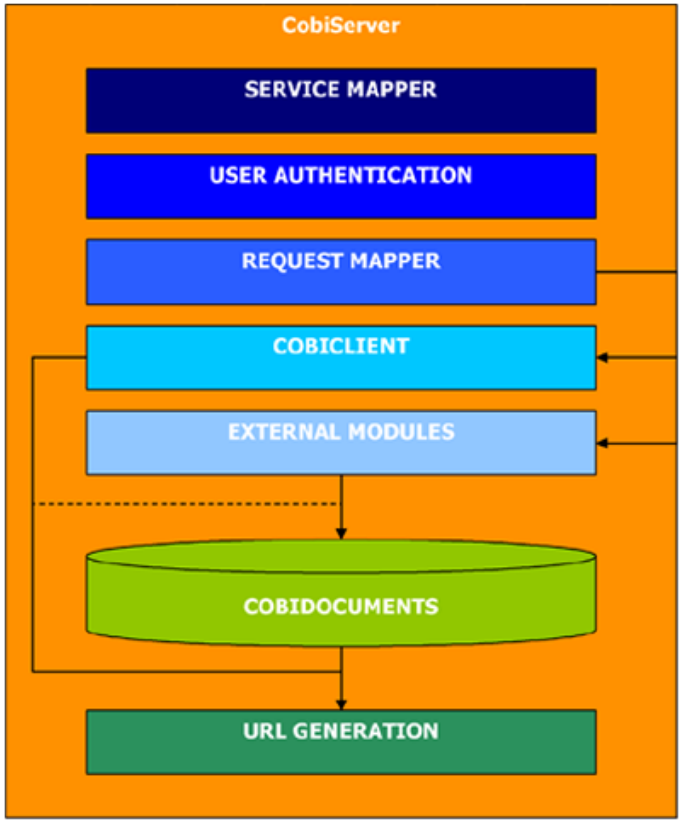


Cobitis client module



The Cobitis client module can query a server to get WSDL and call methods of interest by the way of three specific standard SOAP calls (always exposed by a cobitis server). CobiCall gets method name and a container as parameters. It checks if the method is exposed by one of the server the client is aware of and, if the container elements are consistent with the method requested, it queries the server asynchronously returning results in a container only when the task is completed. The CobiCallURL method works in the same way but it returns as soon as the server responds with the results URL. This URL is valid even if the request is not yet completed on server side and contains information about the task status. It can be fetched using the CobiGetResult method or an HTTP GET URL

Cobitis server module



The server module is based on the cobitis class hierarchy. It behaves as a multi-threaded SOAP server and as a basic multi-threaded HTTP server (allowing WSDL publishing and data retrieving). Its methods can be either local (implemented as cobitis based shared modules) or remote (it contains a copy of the client module). It publishes a WSDL based on available modules and on accessible remote methods. The server can be contacted from any SOAP enabled client. Three specific methods are always exposed and used by the cobitis client to make calls to all other methods (possibly in an asynchronous way) and to retrieve results.

This is of particular interest when long computational tasks are needed. A request mapper checks requested method existence and parameters consistency and decides if the request has to be managed locally or remotely whenever both the options are available. Servers generates URLs pointing to results, allowing both HTTP and SOAP retrieving.

Coding a module

Declare what your module will do...

```
bool COBIMODULEEXPORT CobiGetModule(CobiModule & Module)
{
 bool ret;
 //Add a new group of methods the server will expose
 ret=Module.AddGroup("Sequence analysis");
 //Add a new method
 ret=Module.AddMethod("Sequence analysis","mymethod");
 //Declare input parameters
 ret=Module.AddInputParameter("Sequence analysis","mymethod","seq",ctSequence);
 //And outputs
 ret=Module.AddOutputParameter("Sequence analysis","mymethod","rseq",ctSequence);
 return ret;
}
```

...which function(s) will do it...

```
CobiErrorCode COBIMODULEEXPORT
CobiCallMethod(const char *mname,CobitisContainer &in,CobitisContainer &out)
{
        CobiErrorCode ret=ceUNKNOWNERROR;
        if(!strcmp(mname,"mymethod")) ret=mymethod(in,out);
        return ret;
}
```

...and the function itself

```
CobiErrorCode COBIMODULEEXPORT mymethod(CobitisContainer &in,CobitisContainer &out)
{
 CobiErrorCode ret=ceOK;
 CobitisSequence *seq=(CobitisSequence *)in[0];  //Obtain the first of your parameters...
  ...  //Do something here!
 out.Add(<some cobitis object>);   //Add your results...
 return ret;
}
```

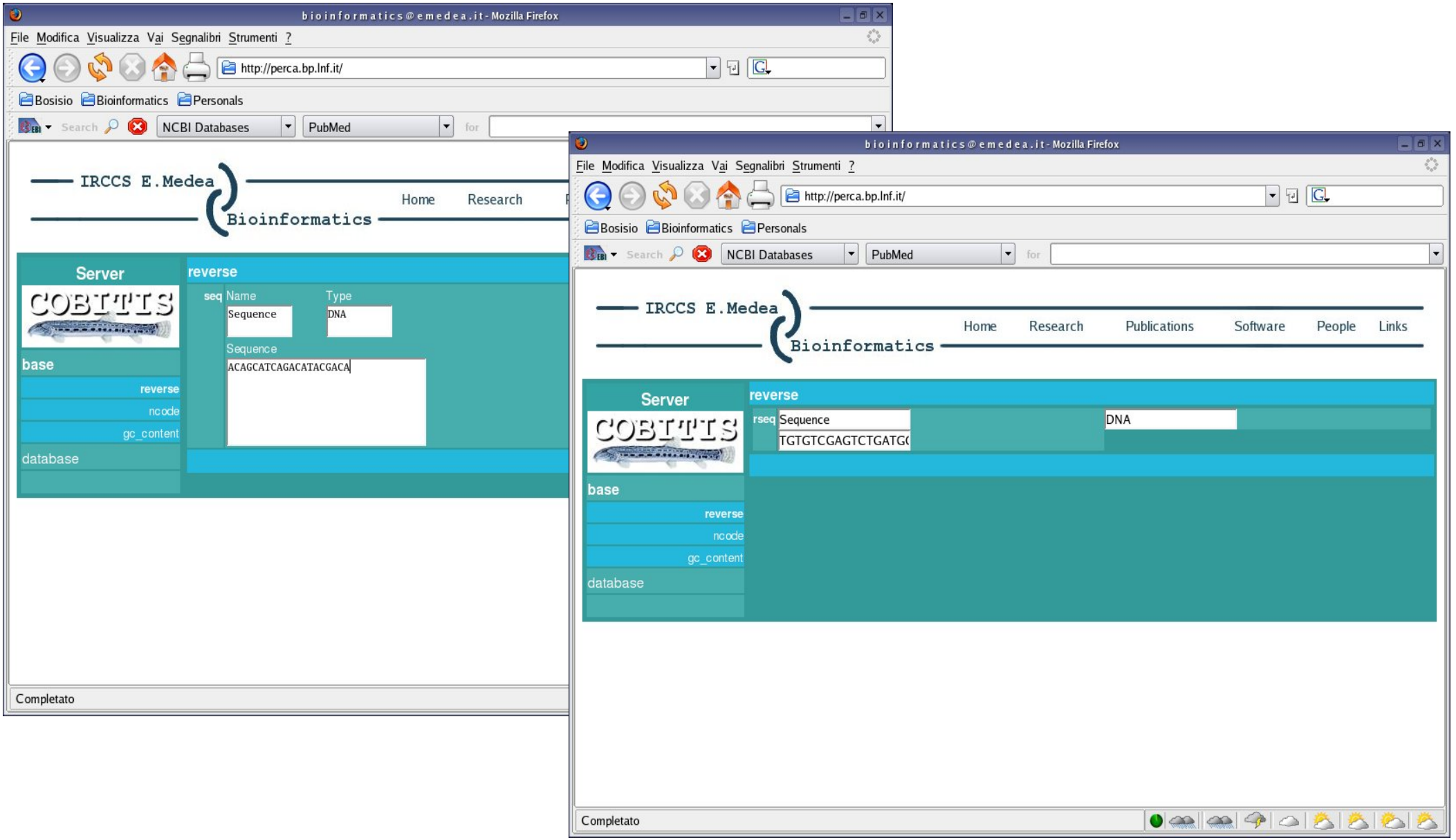
Server configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<cobiconfig:cobiserver xmlns:cobiconfig="http://bioinformatics.emedea.it/cobitis/cobiconfig.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://bioinformatics.emedea.it/cobitis/cobiconfig.xsd" name="perca">
        <system_user name="xxxxx" auth="xxxxxx"/>
        <address address="perca.bp.lnf.it" port="8080"/>
        <paths results="/tmp/cobitis"/>
        <modules>
                <localmodule name="perca" filename="/xxx/xxxxx/xxxxx/cobitis/basemod.so" preloaded="false"/>
                <remotemodule name="rutilus" url="http://rutilus.bp.lnf.it:8080/rutilus.wsdl" public="true"/>
        </modules>
        <groups>
                <group name="base">
                        <method name="reverse"> <endpoint module="rutilus"/> <endpoint module="perca"/> </method>
                  <method name="ncode"> <endpoint module="rutilus"/> </method>
                        <method name="gc_content"> <endpoint module="perca"/> </method>
                </group>
                <group name="database">
                        <method name="mysqlquery"> <endpoint module="perca"/> </method>
                </group>
        </groups>
</cobiconfig:cobiserver>
```

Client configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<cobiconfig:cobiclient xmlns:cobiconfig="http://bioinformatics.emedea.it/cobitis/cobiconfig.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://bioinformatics.emedea.it/cobitis/cobiconfig.xsd http://bioinformatics.emedea.it/cobitis/cobiclient.xsd"
name="test client configuration">
        <proxies>
                <proxy name="xxxxx" address="xxxxxx" port="xx" user="xxxxxx" auth="xxxxxxx"/>
        </proxies>
        <modules>
                <remotemodule name="perca" url="http://perca.bp.lnf.it:8080/perca.wsdl" public="true"/>
                <remotemodule name="barbus" url="http://barbus.bp.lnf.it:8080/barbus.wsdl" public="true"/>
        </modules>
        <groups>
                <group name="Sequence Analysis">
                        <method name="reverse"> <endpoint module="perca"/> </method>
                        <method name="gc_content"> <endpoint module="perca"/> <endpoint module="barbus"/> </method>
                </group>
                <group name="Sequence retrieval">
                        <method name="mysqlquery"> <endpoint module="barbus"/> </method>
                </group>
        </groups>
</cobiconfig:cobiclient>
```

Application examples: php client, web interfaces.



Application examples: console client

Generation of simple XML content:

```
[uberto@perca bin]$ ./cobidoc sequence.seq["Testseq","DNA","acgtacgctagcact"]
<?xml version="1.0" encoding="UTF-8"?><cobitis:document xmlns:SOAP-...
```

Get methods list and they parameters:

```
[uberto@perca bin]$ ./cobiclient -c /etc/cobitis/clientconfig_perca.xml
    Group:  Sequence analysis
        [sequence rseq]=reverse(sequence seq);
        [nummatrix gc_fraction]=gc_content(sequence seq,nummatrix win);
    Group:  Sequence retrieval
        [container query_res]=mysqlquery(charstring query,nummatrix mode);
```

Use of generate content to execute method(s):

```
[uberto@perca bin]$ ./cobidoc sequence.seq["Testseq","DNA","acgtacgctagcact"] |
./cobiclient -c /etc/cobitis/clientconfig_perca.xml -et -m reverse
Mon Oct  3 17:52:37 2005
rseq    [sequence]
>Testseq-DNA
agtgctagcgtacgt
```

Piped clients and xml elements re-mapping make possible methods cascading (workflows through scripts):

```
[uberto@perca bin]$ ./cobidoc sequence.seq["Testseq","DNA","acgtacgctagcact"] |
./cobiclient -c /etc/cobitis/clientconfig_perca.xml -e -m reverse | ./cobiclient -c
/etc/cobitis/clientconfig_perca.xml -et -m reverse seq=rseq
Mon Oct  3 17:58:28 2005
rseq    [sequence]
>Testseq-DNA
acgtacgctagcact
```

Things To Do

Currently are under development:

- An interface between cobitis and R, to make it possible both the exploitation of R as a computational engine (to implement methods) and the use of cobitis from within R itself.
- Further improvement of the php client allowing users to create sessions and to store documents in order to re-use them (or part of them).
- Interfaces between Cobitis and the most popular bioinformatics data formats.
- A module to access MySQL databases.
- A (controlled) extension of the XML scheme.

We plan, in the near future and possibly in collaboration with other people, to develop new interfaces towards other scripting engines extensively used in bioinformatics like (Bio)Perl.