

Soaplab - a unified Sesame door to analysis tools

Martin Senger, Peter Rice, Tom Oinn

European Bioinformatics Institute, Wellcome Trust Genome Campus, Cambridge, UK

<http://industry.ebi.ac.uk/soaplab>

Abstract

Soaplab is a set of Web Services providing programmatic access to many applications on remote computers. Because such applications in the scientific environment usually analyze data, Soaplab is often referred to as an Analysis Web Service. It uses a unified (and partly standardized) API to find an analysis tool, discover what data it requires and what data it produces, to start it and to obtain results. Soaplab is especially well suited for sets of similar tools, such as the European Molecular Biology Open Software Suite (EMBOSS).

Introduction

Do you use command-line tools? Of course, you do... Command-line tools are not only an integral part of any operating system but they play a significant role in any scientific domain. For example, in bioinformatics, there are several sets of command-line tools for analyzing data "in silico", one of the most prestigious of them being the EMBOSS (1).

The advantages of using command-line tools are tremendous: They are fast, efficient and usually easily used by scripting languages. There are, however, also limitations:

- There is no standardisation in the parameters and invocation method for each different tool, thus requiring the user to learn each tool individually.
- Knowledge of the operating system or shell facilities is often required (e.g. how to redirect data flows, or how to call them asynchronously).
- Generally invocation and processing takes place on the same computer.

In spite of these issues, these tools represent an Aladdin's cave of potential treasures for the scientist. One possible approach to overcome their limitations is to write a wrapper around them, making the command-line tools unified, remotely accessible, and hiding their dependencies on the underlying operating system. Soaplab offers such wrappers, and in addition provides for remote invocation of the tools.

Soaplab overview

The Soaplab solution adds complexity but it also removes the fundamental limitations of the command-line tools. It is a suitable approach especially if the task can be summarized by the following facts:

- There is a useful command-line tool or set of tools (such as EMBOSS).
- There is a need to access such tools from other computers, either on a local network or over the global internet.
- There is a requirement to access the tool programmatically, as opposed to manual invocation from the command line or through some kind of CGI interface.

The last point in particular is important considering the growing number of grid-based technologies and the potential orchestration of resources into workflows.

Soaplab is a set of Web Services providing programmatic access to the command-line tools available on remote computers. Because such tools, especially in the scientific environment where Soaplab was born, usually analyze data, Soaplab is often referred to as an Analysis (Web) Service. The important point is that Soaplab services are defined by an API that is the same for all analysis tools, disregarding the operating system where they run, the manner in which they consume and produce data (e.g. from/to files or from/to standard streams), and the precise syntax of the underlying command line tool.

Considering the Soaplab Web Service as a Sesame door to the treasures, what are the keys opening the lock? The answers are in metadata, API and a distributed architecture.

Soaplab metadata

A universal API, as described later, is able to cover all possible command-line tools because Soaplab uses metadata in order to describe individual tools in detail. It is a trade-off allowing a stable API and relatively easy extensibility at the same time. Metadata themselves are extensible (so they can be used not only

for clients but also some implementation may choose to use them to store server-side details there) but the most important facts that are available always include:

- description, type, and provider of the given analysis tool,
- names and types of the input data and command-line parameters,
- names and types of the resulting output data.

Metadata are crucial and service providers (those who want to make their analysis tools available as a Web Service) need to create them for every individual application; for this reason the format of metadata is important. Soaplab stores metadata in XML files - see later about their specification - which is fine for programs but much harder for humans. An example of such XML would be too long to fit into this paper - see it, therefore, online¹.

The service provider needs much easier way. A favourite candidate is the format used by EMBOSS - so called ACD files. It uses simple and forgiving syntax, and - importantly - the EMBOSS 100+ programs are already distributed with their descriptions in this format. Soaplab includes a converter that reads the ACD files and creates the XML file from them. On top of that, the ACD syntax is extensible - it allows a provider to add additional options that are ignored by the official EMBOSS parsers but can be used by the Soaplab generator. The figure 1 shows an ACD file for an EMBOSS *seqret*² program, and the figure 2 shows an ACD file for a mythical application *HelloWorld* (returning just a given greeting).

Soaplab has also additional converters capable of reading other formats and converting them into the unified XML format. Currently absent is a decent graphical editor to create this metadata.

If the metadata are the magic Sesame formula then the Soaplab API is a real key to the Sesame door.

Soaplab API in a nutshell

The main Soaplab API, addressing the individual analysis, allows the client:

- To determine the analysis type, category and all its metadata.
- To send input data and parameters to the analysis.

- To run the analysis - both synchronously (by blocking the request until the analysis finishes) or asynchronously (by creating a session identifier that can be later used by polling the server for the status and results).
- To retrieve current analysis status, including various notification messages (if implemented).
- To retrieve data results.

Figure 3 shows the main API methods (for sake of readability, it is expressed as Java methods but in reality a WSDL is used instead³, Soaplab being a normal Web service).

For feeding data into the analysis and for retrieving results from it the API uses named parameters. The known and allowed names are available from the analysis metadata. The supported types of the input/output data are strings, binary data and arrays of both. Other types can be added by an almost plug-and-play mechanism.

Sometimes it may be more appealing to have a stronger typed API. Such API would not be universal - it would express very specifically methods and data types available only for the given analysis. Soaplab provides it as well: an analysis-specific API can be generated from metadata. This means that any analysis is available (if the service provider decides so) using both a universal API and a strongly typed API (called an API for derived services). Figure 3 shows part of such API for the EMBOSS *seqret* program.

Soaplab provides also a registry-like Web Service that can list all available analysis (from a given server) in a shallow hierarchy. Obviously, this task should be delegated to a proper registry service wherever possible; the functionality is included in Soaplab for the cases where the service provider does not require the more sophisticated facilities of a full registry service such as UDDI. The API of this ill-named service is also in figure 3.

Soaplab architecture

The API and, to certain extent, metadata, are the only things important for the end-users. But for the service providers it is worth to look very briefly into the overall architecture. The figure 4 shows details:

- The part that takes the most effort is to create metadata. These metadata are converted into an XML file and used by the services.

¹ http://industry.ebi.ac.uk/soaplab/copies/blastn_ncbi_al.xml

² <http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/seqret.html>

³ An example:

http://industry.ebi.ac.uk/soaplab/wSDL/edit__seqret.wsdl

- The main block contains the implementation details that can change anytime. The current Soaplab server is based on an internal CORBA-based AppLab (2) server and a Tomcat servlet engine (using Apache Axis Soap toolkit). A less dependency-rich solution is in the pipeline. Note that Soaplab can optionally use a local database for keeping results long after the analysis finished and/or the current client sessions expired; in the absence of this component the local file system is used.
- Thanks to a unified API and the nature of Web Services there can be a range of various clients written in different languages to access the analysis tools remotely. Few of them are distributed with the Soaplab but the more important are those adding values - such as Talisman (3) - a generator of web-based interfaces, or the Taverna project's workbench (4) - a workflow environment. Also some widely used open source initiatives in bioinformatics, such as Bioperl, have their own clients talking to Soaplab-based services already.

Because of having all information defined in metadata, Soaplab is in a very good position to generate many useful things. We have already mentioned generated derived services (Java source code files) with a strongly-typed API, and the converters for generating XML metadata files from other formats. Additionally, Soaplab can automatically generate Java source code files that are able to register some or all Soaplab services by a UDDI registry (using the UDDI4J API), or by BioMoby (5) registry (a work still in progress).

Standards involved

Soaplab services can be considered building blocks (see ^{my}Grid (6) and Taverna project, for example) - and as such they will be more useful the more accepted and open standards they are built on.

The Soaplab API is very similar to the analysis engine interface as defined in the "Biomolecular Sequence Analysis" specification (7) adopted by the Life Sciences Task Force of OMG⁴. The only changes reflect the fact that the original interface was designed purely for CORBA and not for Web services. The same standard defines also the contents of metadata - but only the basics. It allows, however, extension of the metadata in a standard way - which is what Soaplab does.

Having experience with both the API and the metadata, we have initiated a request for a proposal to adopt more middleware-neutral standards. The new specification⁵ is expected at the end of 2003 with the Soaplab to be its prototype.

Conclusion

Having opened the Sesame door, what can be said about the treasures behind? The life science domains are full of cleverly written algorithms and computational methods, very often presented in badly-written scripts with non-existent or limited user interfaces that are sometimes difficult to combine together. Soaplab offers a unified way to access such pearls and provides a basis for integration tools such as Taverna.

An available test bed providing more than one hundred EMBOSS applications is running and accessible at EBI as an experimental service⁶. The whole project was born thanks to the ^{my}Grid project (full acknowledgement in 6), and hopes to evolve further as and when new requirements arise.

⁴ <http://lsw.omg.org>

⁵ <http://www.omg.org/cgi-bin/doc?lifesci/2003-01-08>

⁶ Web Service's endpoint: <http://industry.ebi.ac.uk/soap/soaplab>

Figures

```

appl: seqret [
  documentation: "Reads and writes (returns) sequences"
  groups: "Edit"
]
section: input [ info: "input Section" type: page ]
bool: feature [
  information: "Use feature information"
]
]
seqall: sequence [
  parameter: "Y"
  features: "$ (feature)"
]
]
endsection: input
section: advanced [ info: "advanced Section" type: page ]
bool: firstly [
  information: "Read one sequence and stop"
]
]
endsection: advanced
section: output [ info: "output Section" type: page ]
seqoutall: outseq [
  parameter: "Y"
  features: "$ (feature)"
]
]
endsection: output
  
```

Figure 1: ACD file for an EMBOSS application “seqret”

```


appl: HelloWorld [
  documentation: "Classic greeting
from the beginning of the UNIX epoch"
  groups: "Classic"
  comment: "non-emboss"
  comment: "exe echo"
]
]
string: greeting [
  optional: "Y"
  parameter: "Y"
  default: "Hello World"
  comment: "defaults"
]
]
outfile: output [
  optional: "Y"
  default: "stdout"
]
]
  
```

Figure 2: ACD file for a non-EMBOSS program



Soaplab API

<http://industry.ebi.ac.uk/soaplab/API.html>



The main methods for running an analysis

```

createJob (inputs)
run()
waitFor()
} createAndRun (inputs)
} runAndWaitFor (inputs)

getResults()
destroy()

getSomeResults()
          
```

The List (factory) service

```

getAvailableAnalyses()
getAvailableCategories()
getAvailableAnalysesInCategory()
getServiceLocation()
          
```

The methods for asking about the whole analysis

```

getAnalysisType()
getInputSpec()
getResultSpec()
} describe()
          
```

The methods for asking what is happening

```

getStatus()
getLastEvent()

getCreated()
getStarted()
getEnded()
getElapsed()
} getCharacteristics()
          
```

An example of (additional) methods for a derived service

```

createEmptyJob()

set_sequence_usa (datum)
set_format (datum)
set_firstonly (datum)
...
get_report()
get_outseq()
          
```



Figure 3: Soaplab general API. In the right lower corner there is also a generated strongly-typed API for a particular analysis

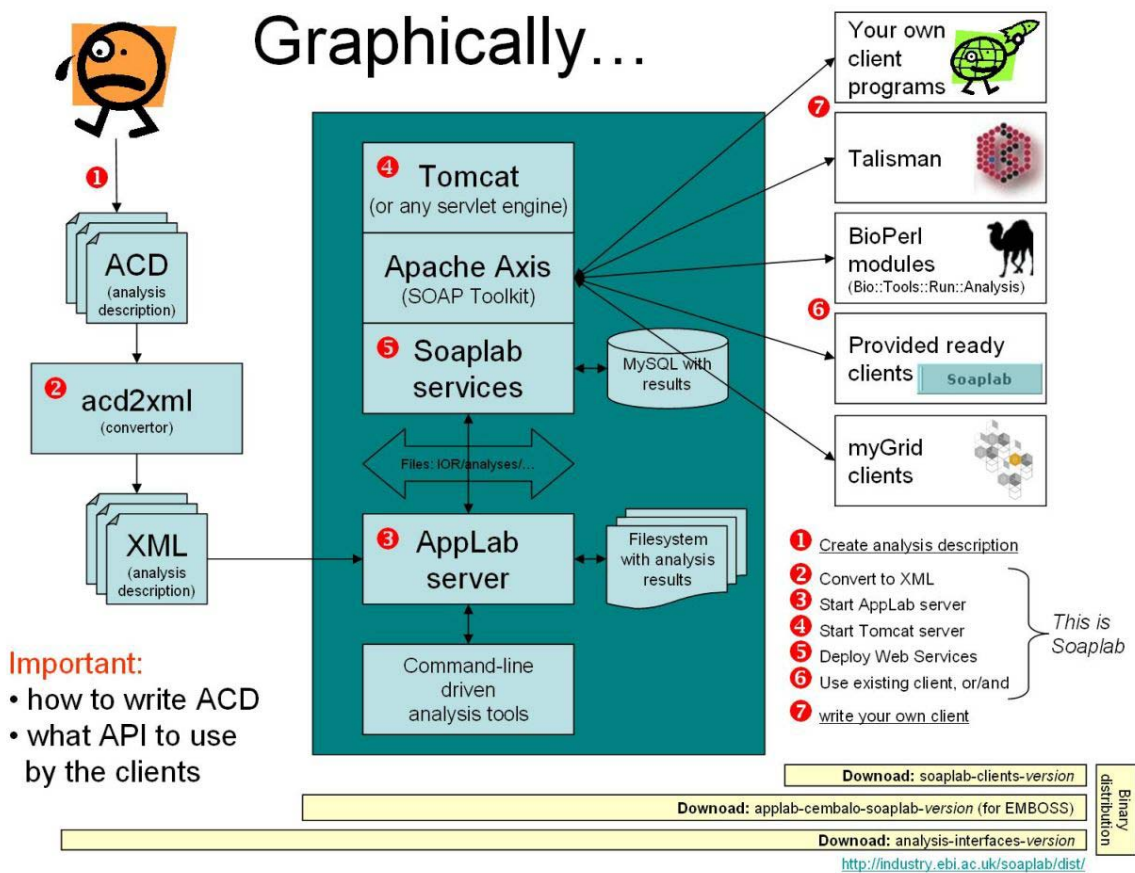


Figure 4: Soaplab Architecture. The main box in the middle represents a “not-so-important” current implementation which may later change without visible impacts on clients.

References

1	Rice,P. Longden,I. and Bleasby,A. "EMBOSS: The European Molecular Biology Open Software Suite" Trends in Genetics June 2000, vol 16, No 6. pp.276-277
2	Senger,M. "AppLab - A CORBA-Java based Application Wrapper" CCP11 Newsletter Issue 8 - 15 June 1999, http://www.hgmp.mrc.ac.uk/CCP11/CCP11newsletters/CCP11NewsletterIssue8.pdf
3	Oinn,T. http://sourceforge.net/projects/jtalisman/
4	Oinn,T. http://sourceforge.net/projects/taverna
5	Wilkinson,MD., Links,M. (2002) "BioMOBY: an open-source biological web services proposal" Briefings In Bioinformatics (2002) 3:4. 331-341.
6	Goble, C.A., Wroe, C.J., Stevens, R. and the myGrid consortium "The myGrid project: services, architecture and demonstrator" Proceedings UK OST e-Science 2nd All Hands Meeting 2003, Nottingham, UK 2-4 Sept, 2003
7	"Biomolecular Sequence Analysis" http://www.omg.org/cgi-bin/doc?dtc/00-11-01