Soaplab - overview

SOAP-based Analysis Web Services

Martin Senger <senger@ebi.ac.uk>

http://www.ebi.ac.uk/soaplab

What is the task?

1. You have (or someone else has) one or more command-line analysis tools (or a relatively simple Web page with interesting contents)
2. You want to access them from other computers, eventually over an Internet
3. You want to access them using your (or someone else's) programs, not just by filling the forms and clicking on web pages

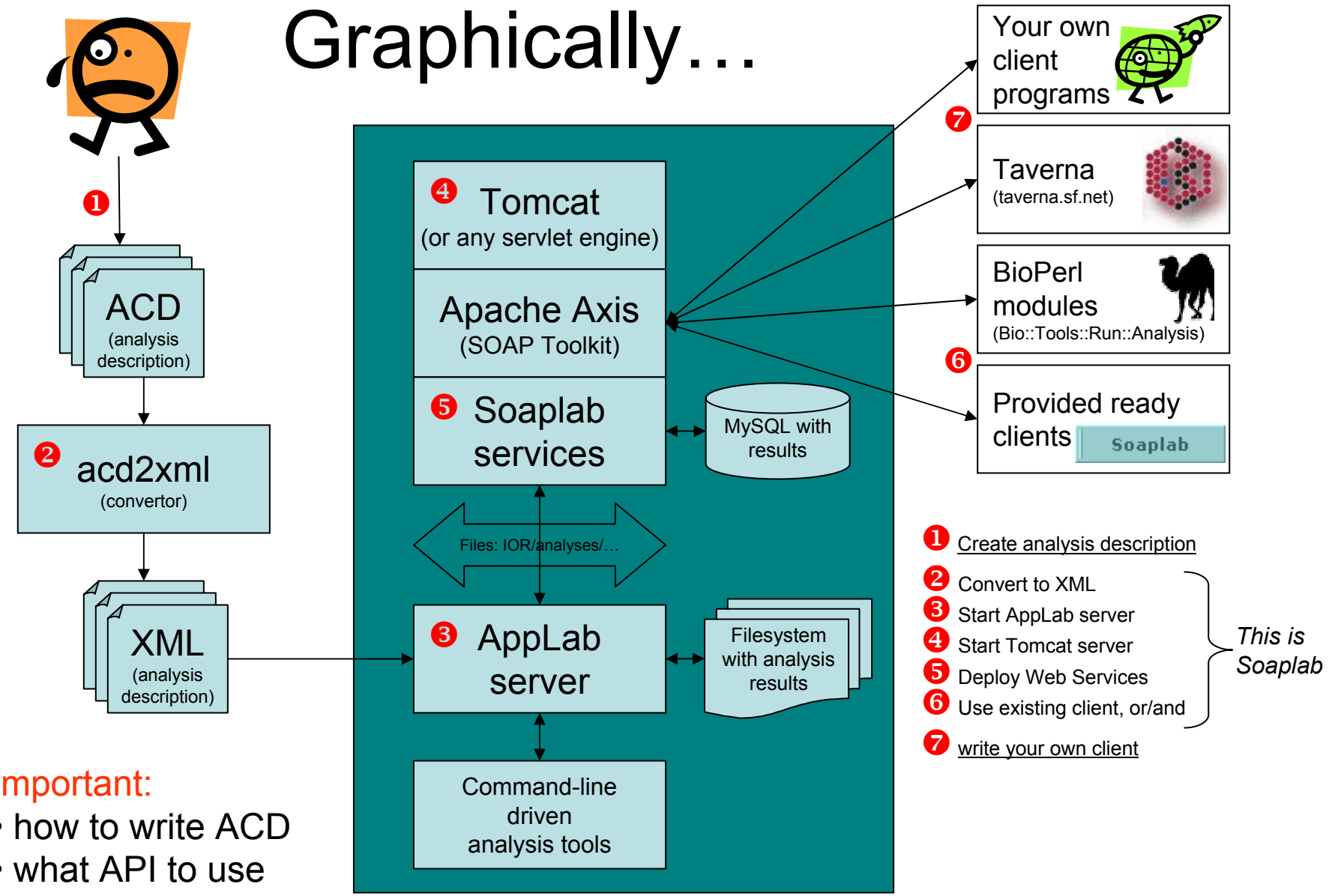
How it fits with the rest of the world

- Phase 1: Create metadata
  - Describe your command-line tool (or a Web page resource), preferable using an ACD format
    - http://emboss.sourceforge.net/developers/acd/index.html
- Phase 2: Use Soaplab
  - Use the description to create and deploy Web Services making your command-line tool (or a web page resource) available
- Phase 3: Write clients
  - Either write your own programs (GUIs and others)
  - Or use Taverna - a workflow system using Soaplab (and other) Web Services as components

What Web Services are in Soaplab?

- By resources that are being accessed:
  - command-line tools (sub-project: AppLab)
  - web pages (sub-project: Gowlab)
- By results that are being delivered:
  - Analysis/Gowlab Factory Service *(misnomen)*
    - produces a list of all available services
  - Analysis Service
    - represents one analysis tool and allows to start it, to control it, and to exploit it, or it represents a remote web page accessible using the same API as other Soaplab analysis services
    - does it in loosely type manner using the same API for all analysis
  - Derived Analysis Service
    - does the same as the Analysis Service above but in a strongly typed manner having individual API for any individual analysis tool

Graphically…



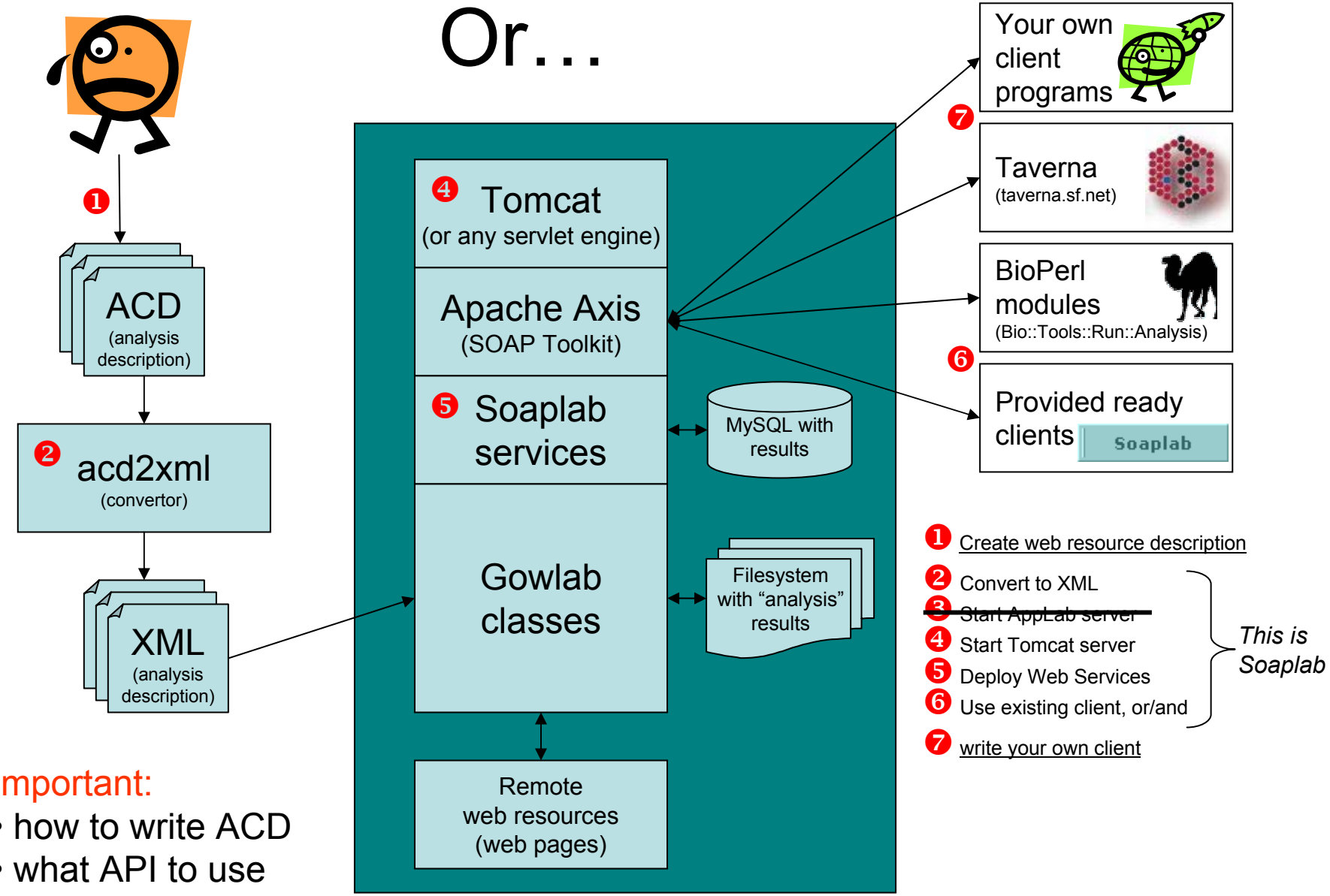
1. Create analysis description
2. Convert to XML
3. Start AppLab server
4. Start Tomcat server
5. Deploy Web Services
6. Use existing client, or/and
7. write your own client

This is Soaplab

Important:

- how to write ACD
- what API to use by the clients

Or…

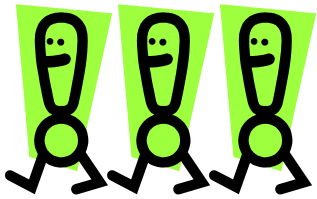


Important:

- how to write ACD
- what API to use by the clients

Soaplab API

http://www.ebi.ac.uk/soaplab/API.html



The main methods for running an analysis

createJob (inputs)
run()
waitFor()

createAndRun (inputs)

getResults()
destroy()

runAndWaitFor (inputs)

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()

Client developer

- Needs to know the Soaplab API
- Can develop in *any* programming language
- Needs to know a place (an endpoint) with a running Soaplab services
  - http://www.ebi.ac.uk/soaplab/services
- Can use the ready, pre-prepared Java and Perl clients, including Taverna

Downoad: soaplab-clients-*version*

http://industry.ebi.ac.uk/soaplab/dist/

Once you know API…


```
import org.apache.axis.client.*;
import javax.xml.namespace.QName;
import org.apache.axis.AxisFault;
import java.util.*;

Call call = null;
try {

    // specify what service to use
    call = (Call) new Service().createCall();
    call.setTargetEndpointAddress ("http://industry.ebi.ac.uk/soap/soaplab/display::showdb");

    // start the underlying analysis (without any input data)
    call.setOperationName (new QName ("createAndRun"));
    String jobId = (String)call.invoke (new Object[] { new Hashtable() });

    // wait until it is completed
    call.setOperationName (new QName ("waitFor"));
    call.invoke (new Object[] { jobId });

    // ask for result 'outfile'
    call.setOperationName (new QName ("getSomeResults"));
    Map result = (Map)call.invoke (new Object[] { jobId, new String[] { "outfile" } });

    // ..and print it out
    System.out.println (result.get ("outfile"));

} catch (AxisFault e) {
    AxisUtils.formatFault (e, System.err, target.toString(),
                           (call == null ? null : call.getOperationName()));
    System.exit (1);
} catch (Exception e) {
    System.err.println (e.toString());
    System.exit (1);
}
```

Once you know API…

- Direct usage of the Soaplab API…

```
use SOAP::Lite;

my ($soap) = new SOAP::Lite
    -> proxy ('http://industry.ebi.ac.uk/soap/soaplab/display::showdb');

my $job_id = $soap->createAndRun()->result;
$soap->waitFor ($job_id);
my $results = $soap->getSomeResults ($job_id, ['outfile'])->result;

print $results->{'outfile'} . "\n";
```

- …or using BioPerl modules

```
use Bio::Tools::Run::Analysis;
print new Bio::Tools::Run::Analysis (-name => 'display::showdb')
    ->wait_for ( {} )
    ->result ('outfile');
```

Soaplab future – what to do

- Possible extensions/improvements
  - Sowa – “Soaplab Without AppLab”
    - reduce complexity on the service provider side
    - allow to re-factor the code to make it more robust
  - Full compliance with the new OMG “Life Sciences Analysis Engine” standard
    - add both-ways notification
    - allow an iterator pattern for huge input/output data

Soaplab future – who will do

- Martin (Soaplab main developer) is leaving EBI on July 2005
- There are at least 3 options regarding further Soaplab development (or their combinations)
  - Martin will be able to allocate some time for Soaplab in his new job
  - Peter Rice's group has resources for further development
  - Martin can be used as a consultant