Soapification

Eric Kow
Laboratoire Loria

25 April 2002

Introduction

What is SOAP?

1. metadata

2. data

3. standards

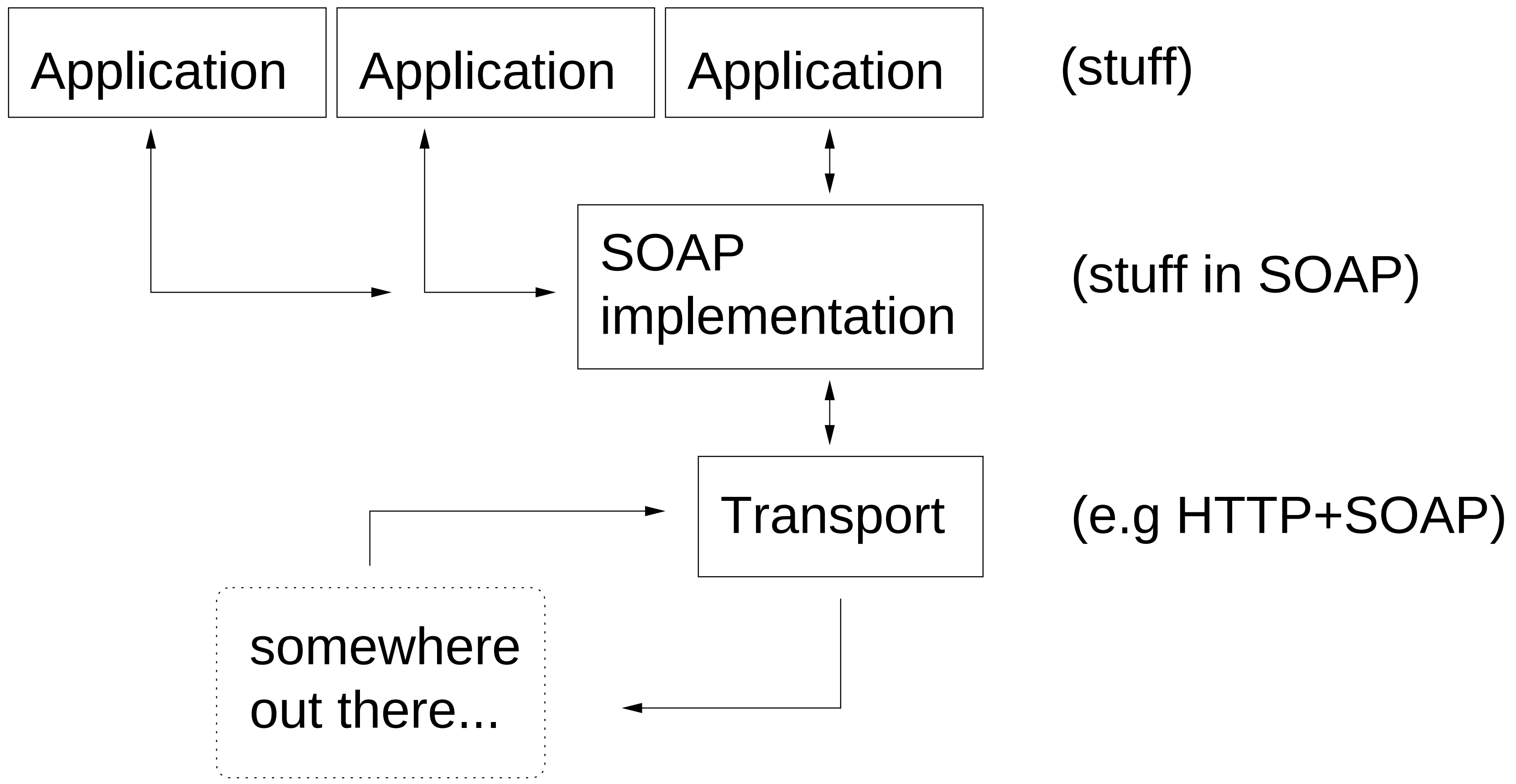
-Single Player-

First we look at what we need to do to make a single module reusable in SOAP.

1. rough idea

2. implementation

Tools (0)


Application
Application
Application
(stuff)
SOAP implementation
(stuff in SOAP)
Transport
(e.g HTTP+SOAP)
somewhere out there...

Tools (1)

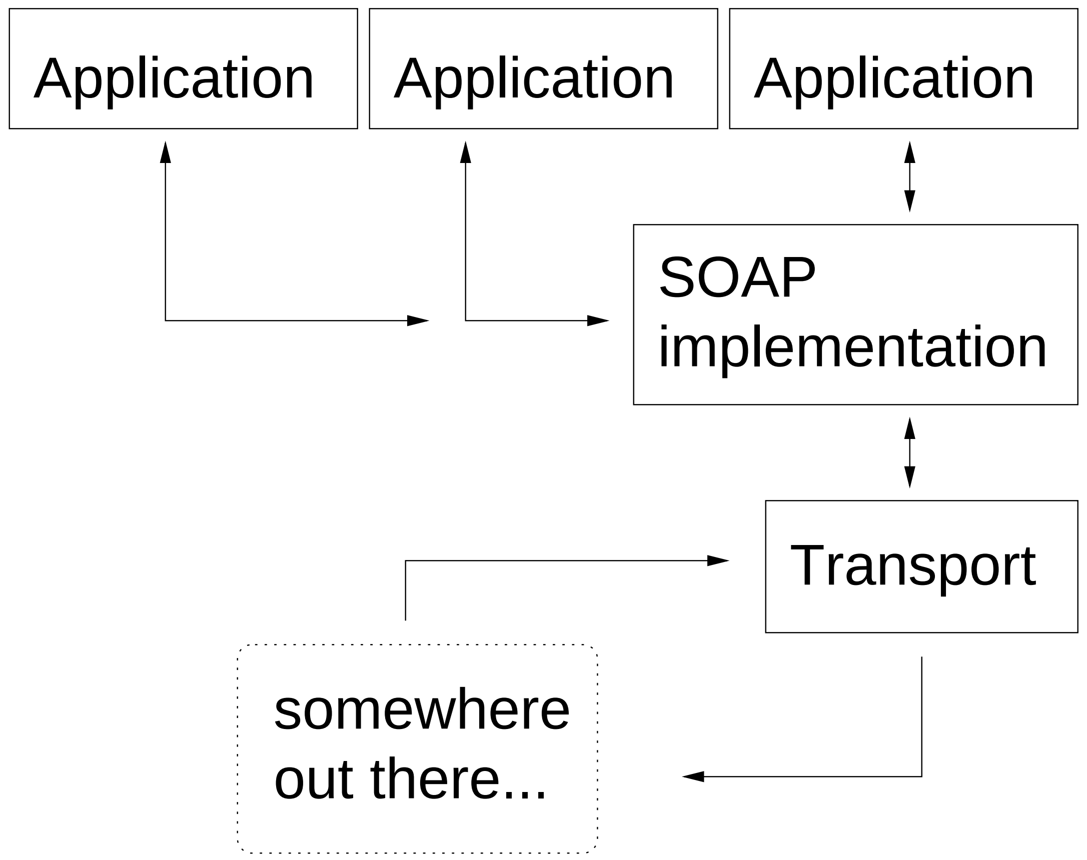
We use Apache SOAP for Java (but there are plenty of others).

You'll want

1. a webserver (like Apache Tomcat)

2. XML implementations (like Xerces)

Tools (2)



PRISM, LexiconService ParserLopez, etc

Apache SOAP 2.2

webserver (Tomcat?)

Your Data

How do we turn data into SOAP (and back)?

- SOAP Encoding
  1. primitive data types (numbers, strings, arrays)
  2. JavaBeans and custom (de)serialisers
- Literal XML (my favourite)

RPC (0)

Apache SOAP implements Remote Procedural Calls. This is the easiest way to use it.

- Server (no work): write a **deployment descriptor** (where on the server your application lives, and what data it gives)

RPC (0)

Apache SOAP implements Remote Procedural Calls. This is the easiest way to use it.

- Server (no work): write a **deployment descriptor** (where on the server your application lives, and what data it gives)
- Client (some work): use Apache SOAP APIs to make RPC calls

RPC (1): Server

What does a deployment descriptor look like?

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:TAGMLService">
  <isd:provider type="java" scope="Application"
                methods="searchLemmaXML getFeatures getTreesXML getWord">
```

RPC (1): Server

What does a deployment descriptor look like?

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:TAGMLService">
  <isd:provider type="java" scope="Application"
                methods="searchLemmaXML getFeatures getTreesXML getWord">
    <isd:java class="fr.loria.led.tagml.TAGMLService"
              static="false"/>
  </isd:provider>
```

RPC (1): Server

What does a deployment descriptor look like?

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:TAGMLService">
  <isd:provider type="java" scope="Application"
                methods="searchLemmaXML getFeatures getTreesXML getWord">
    <isd:java class="fr.loria.led.tagml.TAGMLService"
              static="false"/>
  </isd:provider>

  <isd:faultListener>org.apache.soap.server.DOMFaultListener</isd:faultListener>

  <isd:mappings>
    <!-- here we specify what serialisers/deserialisers
         to use for data that is neither Literal XML
         (org.w3c.dom.Element) or primitive data -->
  </isd:mappings>
</isd:service>
```

RPC (2.1): Client

How about the client end? We first configure the **paramters**.

```
String str = "this is a string";
Element ele = someElement;

Vector params = new Vector();

// this is primitive data handled by SOAP
params.addElement(new Parameter(key, String.class, str,
                                Constants.NS_URI_SOAP_ENC));
// this is raw XML
params.addElement(new Parameter(key, Element.class, ele
                                Constants.NS_URI_LITERAL_XML));
```

RPC (2.2): Client

Secondly, we **construct** and configure the call:

```
private static Call newTAGMLCall(String methodName, Vector parameters) {
  final String service = "urn:TAGMLService";

  SOAPMappingRegistry smr = new SOAPMappingRegistry();

  // Build the call
  Call call = new Call();
  call.setMethodName(methodName);
  call.setParams(parameters);

  call.setSOAPMappingRegistry(smr);
  call.setTargetObjectURI(service);

  // encoding for the data RETURNED by the call
  call.setEncodingStyleURI(Constants.NS_URI_LITERAL_XML);

  // okay, that should be it
  return call;
}
```

RPC (2.3): Client

And finally, we **perform** the call and return the results.

```
private Element performTAGMLCall (Call call) throws SOAPException {
  // Make the call
  Response resp = call.invoke(this.server, "");

  // Check the response.
  if (resp == null) {
    throw new SOAPException(Constants.FAULT_CODE_CLIENT,
                           "Unexpectedly null response from TAGML server");
  } else if (!resp.generatedFault()) {
    Parameter ret = resp.getReturnValue();
    return (Element)(ret.getValue());
  } else {
    Fault fault = resp.getFault();
    throw new SOAPException(fault.getFaultCode(),
                            fault.getFaultString());
  }
}
```

RPC (2.4): Client

Whew! Let's see the end result.

```
public Element getWord(String word) {
  Element wordform;

  // then we perform a call to the server's searchLemma
  try {
    // construct the call
    Vector params = makeStringParams("word",word);
```

RPC (2.4): Client

Whew! Let's see the end result.

```
public Element getWord(String word) {
  Element wordform;

  // then we perform a call to the server's searchLemma
  try {
    // construct the call
    Vector params = makeStringParams("word",word);
    Call call = newTAGMLCall("getWord", params);
```

RPC (2.4): Client

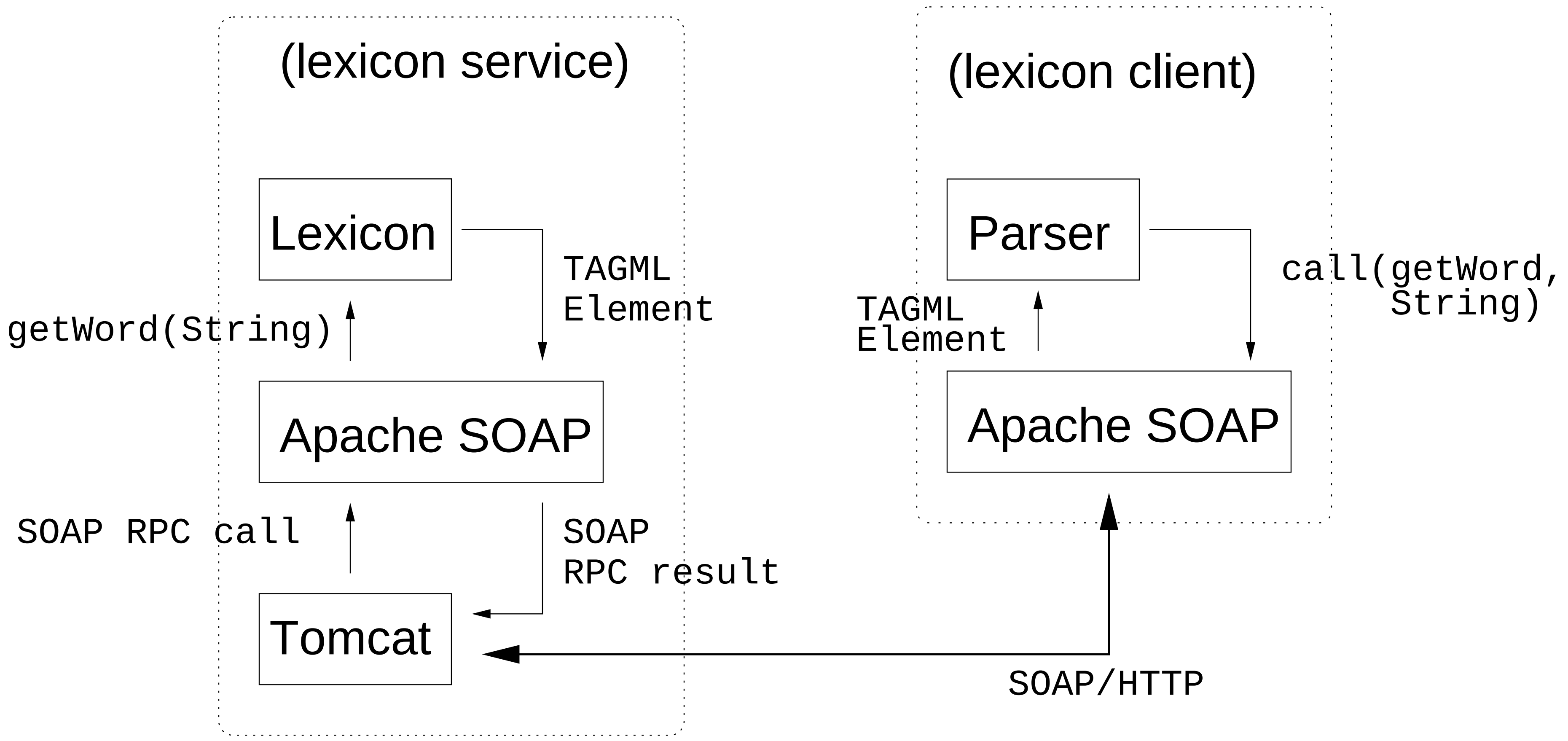
Whew! Let’s see the end result.

```
public Element getWord(String word) {
  Element wordform;

  // then we perform a call to the server’s searchLemma
  try {
    // construct the call
    Vector params = makeStringParams("word",word);
    Call call = newTAGMLCall("getWord", params);
    // Invoke the call and extract the response
    wordform = performTAGMLCall(call);
  } catch (SOAPException e) {
    // FIXME: this should do something intelligent
    System.err.println(e);
    return null;
  }

  return wordform;
}
```

An Example


(lexicon service)
Lexicon
TAGML Element
getWord(String)
Apache SOAP
SOAP RPC call
SOAP RPC result
Tomcat
(lexicon client)
Parser
call(getWord, String)
TAGML Element
Apache SOAP
SOAP/HTTP

Question Marks

There are many things you ought to look into, because i haven't gotten around to them.

1. SOAP messaging instead of RPC (rawer SOAP)

2. other implementations of SOAP, other languages

3. other transport mechanisms (SMTP), other webservers

-MultiPlayer-

How do all these modules fit together? Enter Soapical:

1. Distributed, self-contained modules (bricks)

2. No architectural requirements - every module optional

3. Maximal reuse of generic standards - SOAP, XPath, XLink, etc

Soapical is only a **methodology**, or a convenient way to say all of the above.

Services

We can think of Soapical as a loose collection of services:

1. TagParserService
2. LexiconService
3. TestSuiteService
4. SpeechRecogniserService

Metaservices (0)

We can also think of it as a loose collection of support services:

1. Soapmeter - SOAP message tracking

2. PRISM - XML visualisation

Metaservices (0)

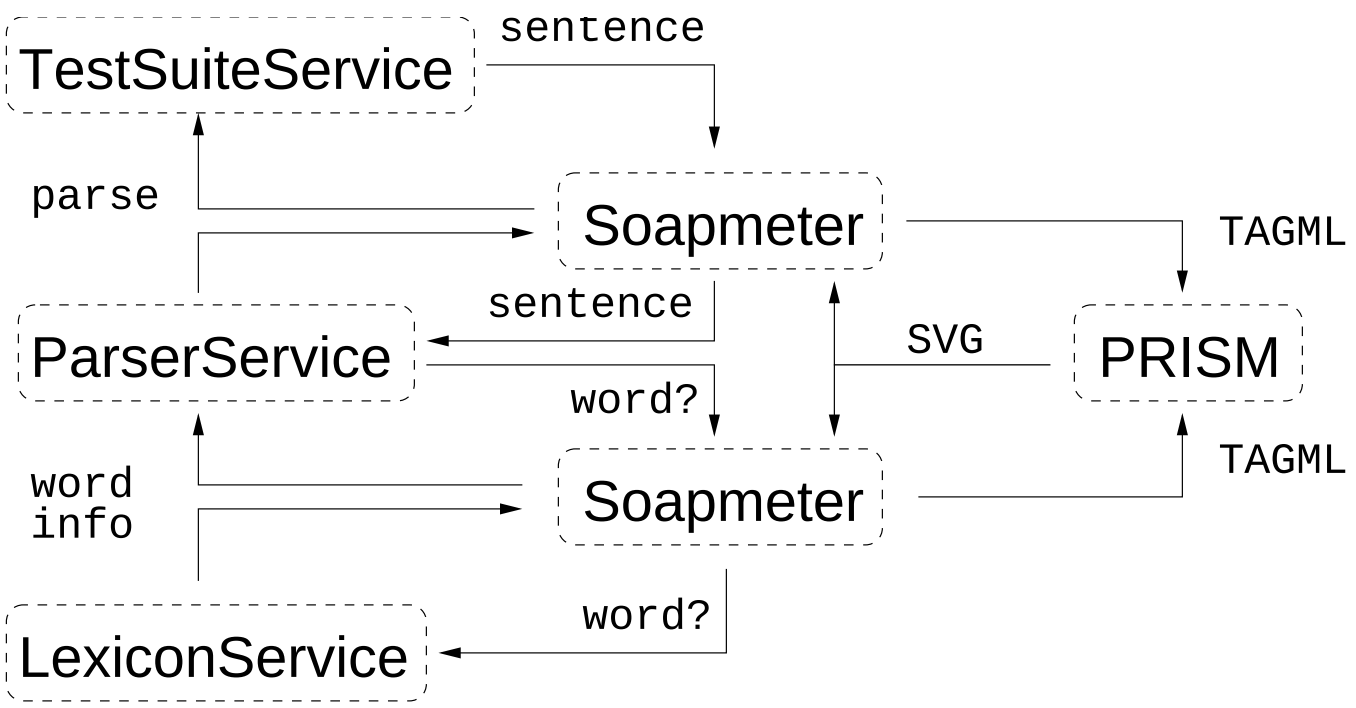
We can also think of it as a loose collection of support services:

1. Soapmeter - SOAP message tracking
2. PRISM - XML visualisation

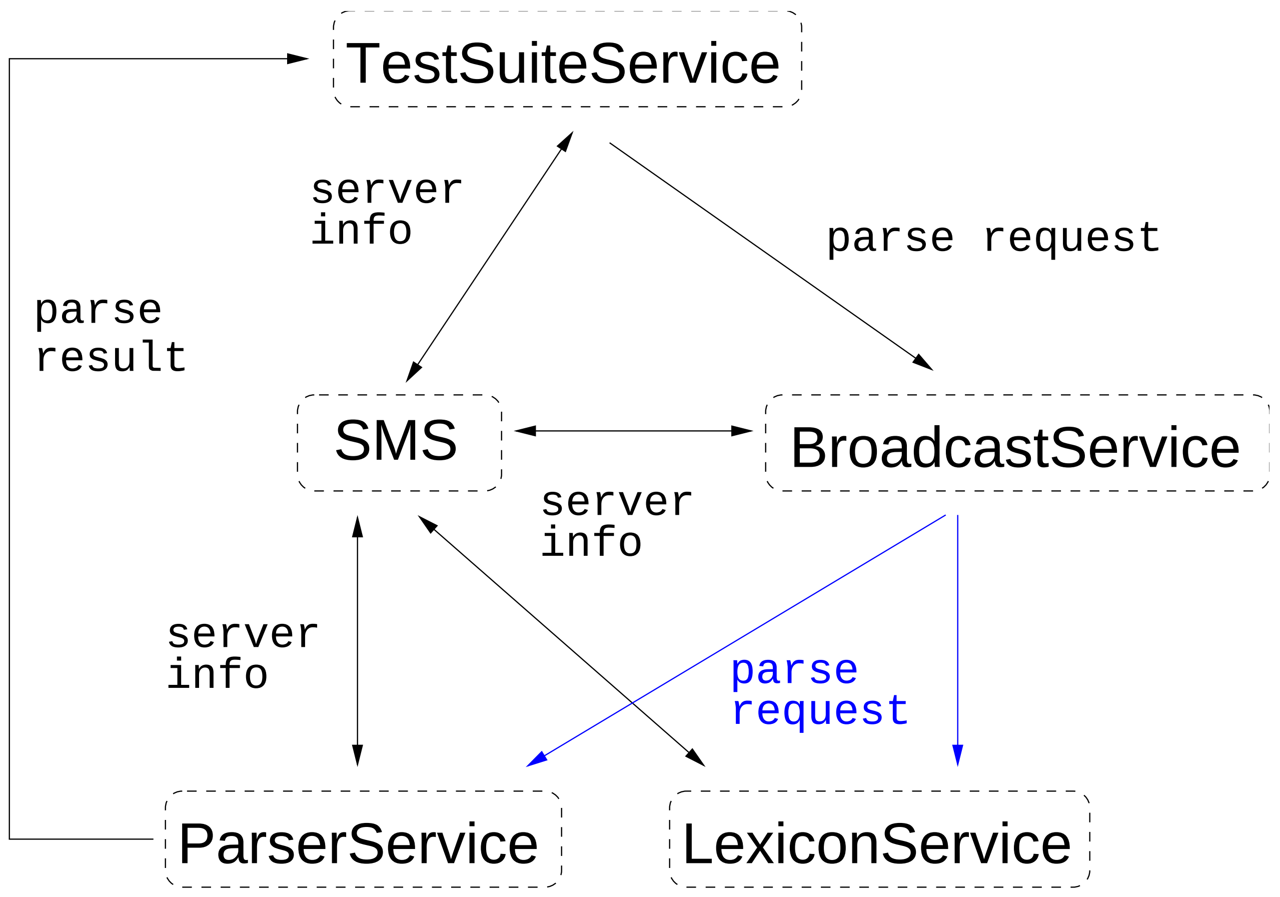
and **optional** architectural services:

1. SMS - keeping track of services
2. BroadcastService - talking to muliple services

Metaservices (1): Support Services



Metaservices (2): Architectural Services



Links (1): Local

Why needlessly repeat bits of data within a single message/document?

```
<miamm:sentence id="42">
   <miamm:word id="1" text="I"/>
   <miamm:word id="2" text="want"/>
   <miamm:word id="3" text="a"/>
   <miamm:word id="4" text="song"/>
   <miamm:word id="5" text="by"/>
   <miamm:word id="6" text="the"/>
   <miamm:word id="7" text="Beatles"/>
</miamm:sentence>
...
<miamm:pos_tag sentence_id="42" word_id="6">noun</miamm:pos_tag>
```

Links (2): Remote

Why send huge chunks of data in the first place (if it's not so likely to be used?)

```
<song_finder:song id="1" title="Blackbird" author="Beatles"
    link="http://songserver.com/986867.mp3/" />
...
<song_finder:search_results>
   <song_finder:search_result song_id="1" score="323"/>
   <song_finder:search_result song_id="4" score="539"/>
</song_finder:search_results>
```

Metadata (0)

Soapical has computational linguists in mind. How do we handle incrementally annotated data?

1. Lots of stuff to do to an utterance text
2. Not all tools do the same job
3. Not all tools do the same job the same way

Metadata (1)

Let's use those SOAP headers!

1. What treatment has been done to this data?

2. What treatment am i looking for?

3. What formats and formalisms am i using?

Danger!

Soapical is only a baby.

1. Soapmeter and PRISM highly imperfect
2. SMS/Broadcast not yet used Soapically
3. We haven't thought through the metadata bits
4. All our work has been with SOAP RPC (not Soapical)