eContent

LIRICS

Deliverable 5.1.E

LIRICS Reference Architecture

Project reference number	e-Content-22236-LIRICS
Project acronym	LIRICS
Project full title	Linguistic Infrastructure for Interoperable Resource and Systems
Project contact point	Laurent Romary, INRIA-Loria 615, rue du jardin botanique BP101. 54602 Villers lès Nancy (France) romary@loria.fr
Project web site	http://lirics.loria.fr
EC project officer	Erwin Valentini
Document title	LIRICS Reference Architecture
Deliverable ID	**D5.1.E**
Document type	Report
Dissemination level	Public
Contractual date of delivery	M24
Actual date of delivery	7 Feb 2007
Status & version	Draft
Work package, task & deliverable responsible	WP5, USFD
Author(s) & affiliation(s)	Julien Nioche, Adam Funk (USFD)
Additional contributor(s)	
Keywords	LIRICS Reference Architecture

Document evolution

version	date	version	date
0.1	25/01/06		
1.0	30/06/06		
2.0	07/02/07		

Table of contents

1 Introduction

This document describes the architecture of the reference implementation of LIRICS.

This reference implementation consists of a set of open-source, web service applications that carry out various types of linguistic analyses on natural language texts in accordance with the relevant standards developed in LIRICS and in association with it. These applications are useful in their own right but also serve to demonstrate the practical usability of the LIRICS standards.

Each application is deployed according to well-known web service standards (exchanging XML messages through the SOAP protocol) in order to provide the following benefits:

- interoperability between domains, applications, and users;
- composability of operations (for example, users can program their clients to send the output from one service as input to another);
- encapsulation and abstraction (the users do not need to be concerned with the details “behind the scenes”, i.e. how the services work internally).

The reference implementation will be provided for the DCR (Data Category Registry), LMF (Lexical Markup Framework), MAF (Morphosyntactic Annotation Framework) and SynAF (Syntactic Annotation Framework) standards, but not for the Semantic Annotation standard.

2 DCR implementation

In order to allow external applications to access the Data Category Registry to browse, search and select datacategories from the DCR a web based interface has been developed to allow for these usage scenario's. A short introduction of the covered use cases is supplied, followed by a more detailled discussion of the web based service interface.

2.1 Use case summary

The following use cases have been specified and implemented on the SYNTAX DCR server

- Browse catalogue(basic browsing)
    - Standard navigation over the DCR is done by selecting a profile of interest after which a list of data categories are retrieved from the DCR that are part of this profile. Optionally, a registration status may be specified to limit the number of data categories to be retrieved from a profile to only those with the specified registration status.

- Search catalogue
    - The DCR may be also searched by specifying the search terms and optional parameters such as profile to search or data category sections(title, description etc) to search. An overview of the interaction is shown below.

2.2 Implementing DCR access

The interface interactions described above have been implemented on the SYNTAX server using a web based service interface (REST). Operations are accessible via parameterized HTTP requests, results are delivered in xml messages. Additionally a java package has been developed at the MPI shielding the developer from the intricacies of service request modelling and xml message deserialization. Calls are made over a standard java interface and results are delivered as standard java objects. This approach has been used in connecting LEXUS to the DCR server.

The following operations are available.

- GetProfiles()
    - Returns the list of profiles for the DCR
    - http://syntax.inist.fr/mod_webservice/call.php?fct=getProfiles
- getDataCategories(a_profile, a_registrationStatus)
    - Returns the data categories associated with the specified profile. The registrationStatus is optional and filters out only the data categories matching the specified registrationStatus.

    - http://syntax.inist.fr/mod_webservice/call.php?fct= getDataCategories& profile= *a_profile* & status= *a_registrationStatus*

- getDataCategory(a_urid)

    - Returns the data category identified by the specified urid

    - http://syntax.inist.fr/mod_webservice/call.php?fct= getDataCategory& urid= *a_urid*

- searchDataCategories(a_listOfKeywords, a_listOfFields, a_profile, a_registrationStatus)

    - Returns the data categories matching the specified parameters

    - http://syntax.inist.fr/mod_webservice/call.php?fct= searchDataCategories& keywords[]= *a_keywords* & fields[]= *a_fields* & profile= *a_profile* & status = *a_registrationStatus*

3 MAF implementation

The MAF reference implementation for English and Bulgarian is described here. It has been implemented as a Web Service, reusing the existing resources for GATE. The reference implementation will be provided by the deliverable D5.2.C.

The reference implementations for French (INRIA), Italian (CNR), Spanish and German (DFKI) will be provided by the respective partners.

3.1 Description

The MAF service for English is available from (http://gate.ac.uk/lirics/MAFservice). It is based on the following GATE processing resources:

- ANNIE English Tokenizer
- ANNIE Sentence Splitter
- ANNIE POS Tagger
- GATE Morphological Analyser
- GateAnnots2MafAnnots

The Part of Speech tagger has been developed by Mark Hepple and is released with GATE. It uses a slightly modified version of the PennTreebank tagset (http://www.gate.ac.uk/sale/tao/index.html#x1-368000D).

The morphological analyser generates information about the lemma of words.

The *GateAnnots2MafAnnots* PR converts the POS tags generated by the tagger into DCR values. It requires a mapping file and a XML representation of the MAF tagset. Note: there is no access to the DCR: the values are set once and for all.

As specified in the deliverable D5.1.C “API for morpho-syntactic annotations”, a MAF compliant service takes as input some text and returns a MAF XML document. The reference implementation for English and Bulgarian creates a GATE document with the text provided by the client of the service and runs the processing resources described above. As a result, the service has a GATE document representing the content of a MAF document. A XML representation of the document is then generated and returned to the client application.

The reference implementation for Bulgarian uses a different Part of Speech tagger. The Bulgarian Tree Bank project has provided us with an annotated corpus, which has been used to generate a statistical model for the TreeTagger (http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html). This model has been made available to the research community on the TreeTagger website. GATE has a plugin for the TreeTagger, which will be used instead of the default ANNIE tagger for the Bulgarian. The TreeTagger has already some models for French, German, Italian, English and Spanish, which means that it would be possible to provide a MAF reference implementation for these languages using the solution based on GATE. This would only require a mapping file from the tagsets used by the tagger to DCR values. The mapping file could be used by the GateAnnots2MafAnnotsPR. The MAF documents generated for the Bulgarian do not have any information about the lemmas.

documents

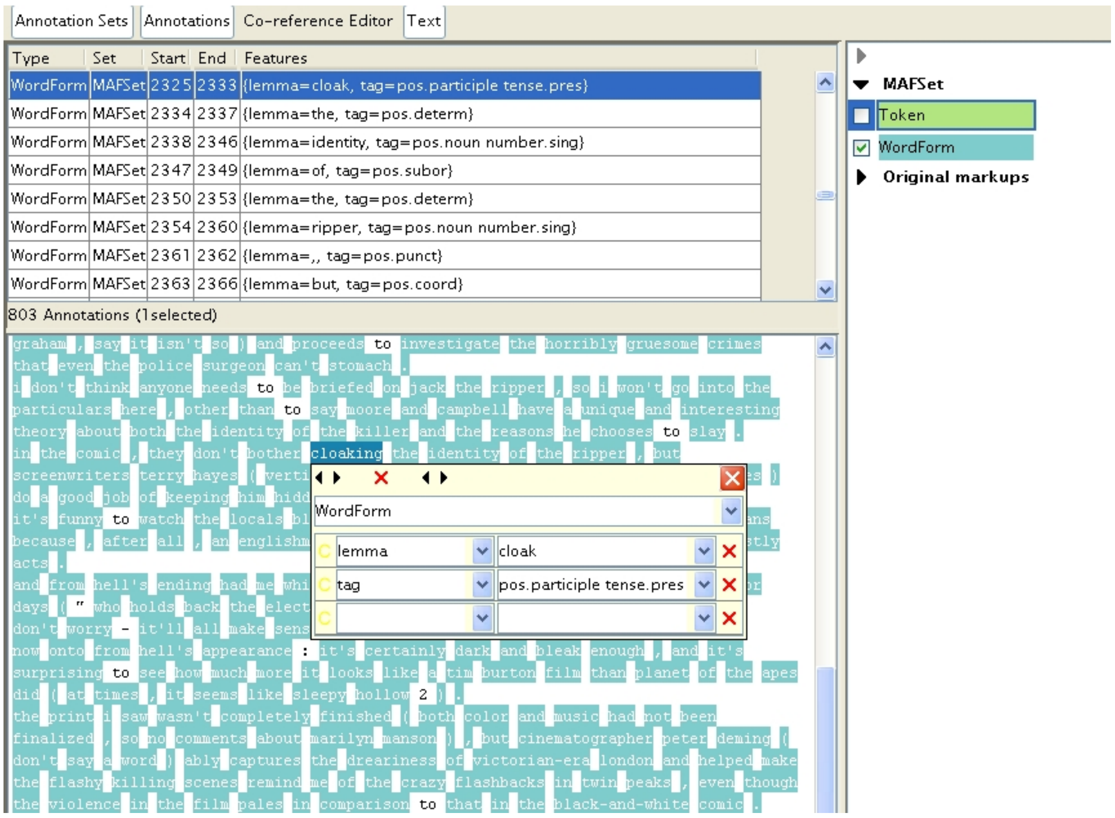
3.2 Representing a MAF document in GATE

The reference implementation for English and Bulgarian consists of a web service using GATE to generate Part of Speech annotations for a text. As explained above, a GATE document is used to represent the information contained in a MAF document. Such a MAF GATE Document has a few specificities:

1. The document has a feature named **tagset** which is a xml representation of the tagset, as specified in the MAF specification

2. The document contains two types of annotations, **Token** and **Wordform**, corresponding to the respective elements in MAF. The **wordform** has a feature named **tag**, the value of which must be specified in the **tagset** as well as a feature **lemma**.

The **Figure** 1 shows a MAF document in GATE.



Type	Set	Start	End	Features
WordForm	MAFSet	2325	2333	{lemma=cloak, tag=pos.participle tense.pres}
WordForm	MAFSet	2334	2337	{lemma=the, tag=pos.determ}
WordForm	MAFSet	2338	2346	{lemma=identity, tag=pos.noun number.sing}
WordForm	MAFSet	2347	2349	{lemma=of, tag=pos.subor}
WordForm	MAFSet	2350	2353	{lemma=the, tag=pos.determ}
WordForm	MAFSet	2354	2360	{lemma=ripper, tag=pos.noun number.sing}
WordForm	MAFSet	2361	2362	{lemma=,, tag=pos.punct}
WordForm	MAFSet	2363	2366	{lemma=but, tag=pos.coord}

803 Annotations (1selected)

```
graham , say it isn't so ) and proceeds to investigate the horribly gruesome crimes
that even the police surgeon can't stomach .
i don't think anyone needs to be briefed on jack the ripper , so i won't go into the
particulars here , other than to say moore and campbell have a unique and interesting
theory about both the identity of the killer and the reasons he chooses to slay .
in the comic , they don't bother cloaking the identity of the ripper , but
screenwriters terry hayes ( verti                                            es )
do a good job of keeping him hidd
it's funny to watch the locals bl                                          ans
because , after all , an englishm                                          stly
acts .
and from hell's ending had me whi                                        or
days ( " who holds back the elect
don't worry - it'll all make sens
now onto from hell's appearance : it's certainly dark and bleak enough , and it's
surprising to see how much more it looks like a tim burton film than planet of the apes
did ( at times , it seems like sleepy hollow 2 ) .
the print i saw wasn't completely finished ( both color and music had not been
finalized , so no comments about marilyn manson ) , but cinematographer peter deming (
don't say a word ) ably captures the dreariness of victorian-era london and helped make
the flashy killing scenes remind me of the crazy flashbacks in twin peaks , even though
the violence in the film pales in comparison to that in the black-and-white comic .
```

WordForm

lemma	cloak
tag	pos.participle tense.pres

Figure 1 a MAF document in GATE

The Lirics Platform being based on GATE, the document processed by a remote MAFService will be represented as a MAF GATE document and might be passed to another LIRICS reference implementation such as SYNAF or processed in a local application.

An intermediate API has been developed by the University of Sheffield for representing a MAF document, independently from GATE. This is described as the low level API for MAF. The aim of this API is to represent a MAF document, generate it from a MAF XML but also generate a XML for a MAF document.

A native implementation of MAF (i.e not based on a web service) would provide an implementation of this low level API in order to manipulate the content of a MAF document using a language like Java.

In the reference implementation the low level API is used in order to generate a XML representation from a MAF GATE document. The XML representation is returned to the client by the web service.

3.3 Compiling and installing the reference implementation

The reference implementation is available as a set of Java classes. In order to use or compile the code, you should install Java (http://java.sun.com/) and ANT (http://ant.apache.org/). The web service can be implemented using Tomcat (http://tomcat.apache.org/).

The code is organised by packages:

- **lirics.maf**: low level MAF API as a set of interfaces, which would be implemented by some solution provider.
- **lirics.maf.impl** : an implementation of the MAF low level API
- **lirics.maf.xml** : de/serialization of a MAF object from/to MAF XML
- **lirics.maf.gate** : GATE resources for MAF

The sources also contain a couple of web service related packages:

- **lirics.service**
- **lirics.maf.service.client.maf**

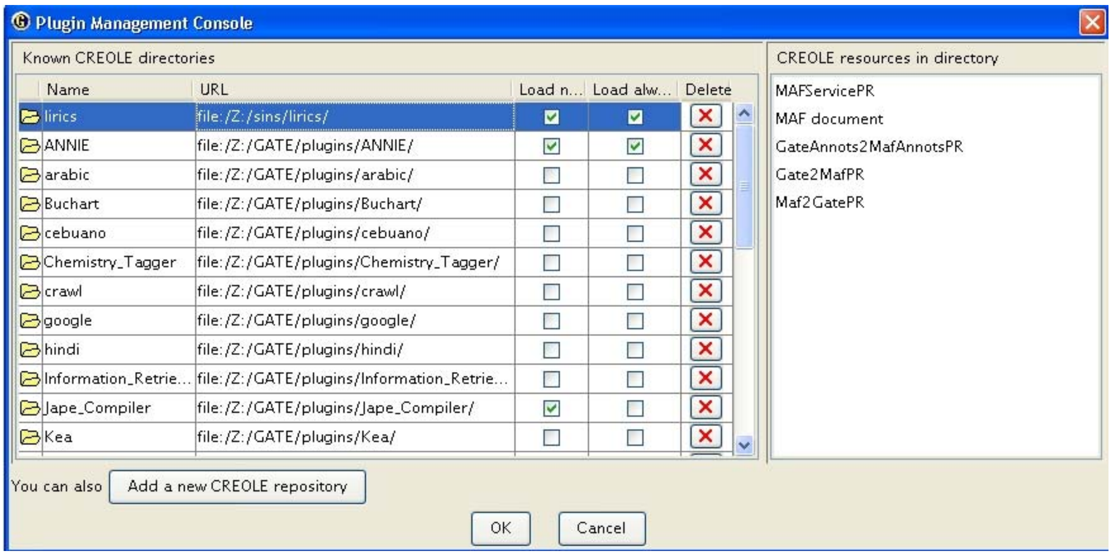
The latter contains a set of classes that can be used inside a Java program in order to communicate with a MAF web service.

The code can be compiled from the root directory of LIRICS by typing ‘ant’. A file named *lirics.jar* will be automatically generated.

3.4 Testing the reference implementation

The reference implementation can be tested using the classes located in the package **lirics.maf.service.client.test.**

Another way to test the implementation is to load the LIRICS plugin for GATE. For that, load the CREOLE plugin console of GATE (indicated by <?>), click on “*add creole repository*” and enter the location of the LIRICS directory. The LIRICS plugin will be added to the list of available plugins, the panel on the right will show all the LIRICS Resources available for GATE, as shown on the **Figure** 2.


Plugin Management Console

Known CREOLE directories

Name	URL	Load n...	Load alw...	Delete
lirics	file:/Z:/sins/lirics/	✓	✓	✗
ANNIE	file:/Z:/GATE/plugins/ANNIE/	✓	✓	✗
arabic	file:/Z:/GATE/plugins/arabic/	☐	☐	✗
Buchart	file:/Z:/GATE/plugins/Buchart/	☐	☐	✗
cebuano	file:/Z:/GATE/plugins/cebuano/	☐	☐	✗
Chemistry_Tagger	file:/Z:/GATE/plugins/Chemistry_Tagger/	☐	☐	✗
crawl	file:/Z:/GATE/plugins/crawl/	☐	☐	✗
google	file:/Z:/GATE/plugins/google/	☐	☐	✗
hindi	file:/Z:/GATE/plugins/hindi/	☐	☐	✗
Information_Retrie...	file:/Z:/GATE/plugins/Information_Retrie...	☐	☐	✗
Jape_Compiler	file:/Z:/GATE/plugins/Jape_Compiler/	✓	☐	✗
Kea	file:/Z:/GATE/plugins/Kea/	☐	☐	✗

CREOLE resources in directory

- MAFServicePR
- MAF document
- GateAnnots2MafAnnotsPR
- Gate2MafPR
- Maf2GatePR

Figure 2 GATE plugin management console

The **MAFServicePR** will provide an access to a remote MAF service and can be used inside a normal GATE application (see GATE documentation).

MAFDocument is not a Processing Resource, but a Language Resource. It gives the possibility to load a MAF XML document inside GATE and represent it as a MAF GATE Document (see chapter 3.2 **Representing a MAF document in GATE**).

The deliverable D5.3.B describes with more details the LIRICS integration platform.

4 LMF implementation

The LMF reference implementation is based on the functionality provided by the LEXUS tool. LEXUS's functionality includes creation, reading, updating and removal of LMF related items and implements the construction mechanism proposed by the ISO/TC 37/SC 4 drafts. Advanced search functionality is provided allowing users to search multiple heterogeneously structured lexical resources simultaneously.

To provide the LMF web service functionality it has been decided to open up LEXUS's core functionality to allow operations using a web services interface. Specification of this interface has been done in earlier work as part of this project. Implementation of this interface however has been delayed due to the fact that a suitable xml exchange format for LMF lexica is currently still under development. The proposed structure as presented in the LMF ISO document (Annex R of the ISO/TC 37/SC 4 Rev. 13) is not capable of expressing the diversity in structural elements in lexica as are encountered within the MPI and other organizations. In essence it only allows for bundling of lexica that share a common structure. For exchange purposes in an environment where lexica are structured in various ways, such as in MPI's archive, a different approach is needed. This issue is currently being addressed. Since the goal of the web service interface is to provide access to LEXUS's functionality and information exchange will be message based, i.e. information content is transferred as xml fragments, it is essential that this issue is resolved before implementation of the web service interface can commence.

4.1 Description

The proposed web service will be based on LEXUS's functionality which is available at http://www.mpi.nl/mpi/lexus.

LEXUS's core functionality consists of the following:

- Flexible lexicon schema creation.

  Lexica may be structured according to the researcher's needs, i.e. language or theoretical approach.

- Integration of standard data category definitions from data category registries, such as ISO 12620.

  Data categories may be user defined or may be selected from well established standard concept registries.

- Easy manipulation of lexicon content.

  Lexicon content may be crated, modified and removed within structural constraints.

- Insertion of multimedia content

  Multimedia content (audio, images, video, etc) may be added to lexical entries to create rich multimedia content

- Structural integrity

  Integrity of the lexicon structure is adhered to for all content, i.e. lexical entries, of the lexicon.

- Advanced search capabilities.

  Multiple differently structured lexica may be searched simultaneously to allow for easy comparison and lookup.

- User defined views.

  Views on lexical entries are completely customizable by users, both in content to be displayed as well as look and feel.

The web service’s functionality is based on some of the core functionality of LEXUS, mainly schema extraction and lookup facilities.

5 SynAF implementation

The SynAF reference implementation for English is described here. It will be implemented as a Web Service, largely reusing existing resources for GATE. The reference implementation will be provided by deliverable 5.2.C.

Reference implementations for Bulgarian, French, German, Italian and Spanish will also be provided.

5.1 Description

The SynAF service for English will be available as a web service from the host gate.ac.uk. It will be based on the following GATE components:

- ANNIE English Tokenizer
- ANNIE Sentence Splitter
- ANNIE POS Tagger
- GATE Morphological Analyser
- GateAnnots2SynafAnnots

as described in Sectioon 3.1, as well as a suitable parser to be selected as part of our development of this implementation. The GateAnnots2SynafAnnots PR will convert the syntactic tags generated by the parsers into DCR values, based on a mapping file and an XML representation of the SynAF tagset.

As specified in deliverable D5.1.D (“API for syntactic annotations”), a SynAF service takes as input a natural-language text (either unannotated or annotated in accordance with the MAF specification) and returns a SynAF XML document.

Internally, the SynAF reference implementation will generate a GATE document containing the input text (and if appropriate, the MAF annotations) supplied by the client, and will run the processing resources listed above, to produce a GATE document representing the content of a SynAF document; it will then generate an XML representation, which it will return to the client.

5.2 Representing a SynAF document in GATE

The reference implementation will use a GATE document to represent a SynAF document in a similar manner to that described for the MAF reference implementation in Section 3.2. The specifications will be determined as part of the development of the implementation and will be explained in detail in subsequent deliverables.

As in the MAF case, a native (non-web-service) implementation of SynAF will allow programs to manipulate SynAF documents through a low-level API.