LIRICS

Deliverable 5.3.B

LIRICS Integration Platform

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	**D5.3.B**
Document type	Report
Dissemination level	Public
Contractual date of delivery	M18
Actual date of delivery	30 June 2006
Status & version	Draft
Work package, task & deliverable responsible	WP5, USFD
Author(s) & affiliation(s)	Julien Nioche (USFD)
Additional contributor(s)	Adam Funk (USFD)
Keywords	LIRICS Reference Architecture

Document evolution

version	date	version	date
0.1	09/02/06		
1.0	30/06/06		

Introduction

In order to demonstrate how LIRICS standards facilitate the building of multi-component NLP systems and, more importantly, to provide users with means to use LIRICS-compliant distributed resources and build more complex NLP applications based on them, an open-source LIRICS Service Integration Platform is provided which is based on GATE. This platform will help users with LIRICS service composition, process flow, and debugging. The current document describes the LIRICS service platform.

1 Overview of GATE

Rather than starting from scratch, we chose to build on the world-class open-source GATE infrastructure (http://gate.ac.uk). GATE has already a set of graphical tools that support the creation, execution, and debugging of traditional NLP applications, i.e., applications consisting of components executed on the same machine. Therefore, GATE's component model will be extended to support distributed LIRICS services and a new application flow editor will be implemented that enables service composition and debugging.

GATE can be defined as:

- An *architecture* describing how language processing systems are made up of components.
- A *framework* (or class library, or SDK), written in Java and tested on Linux, Windows and Solaris.
- A *graphical development environment* built on the framework.

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of component, known as resources. Components are reusable software chunks with well-defined interfaces, and are a popular architectural form, used in Sun's Java Beans and Microsoft's .Net, for example. GATE components are specialised types of Java Bean, and come in three flavours:

- LanguageResources (LRs) represent entities such as lexicons, corpora or ontologies;
- ProcessingResources (PRs) represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers;
- VisualResources (VRs) represent visualisation and editing components that participate in GUIs.

When using GATE to develop language processing functionality for an application, the developer uses the development environment and the framework to construct resources of the three types. This may involve programming, or the development of Language Resources such as grammars that are used by existing Processing Resources, or a mixture of both. The development environment is used for visualisation of the data structures produced and consumed during processing, and for debugging, performance measurement and so on. For example, figure 1 is a screenshot of one of the visualisation tools displaying named-entity extraction results for a Hindi sentence.

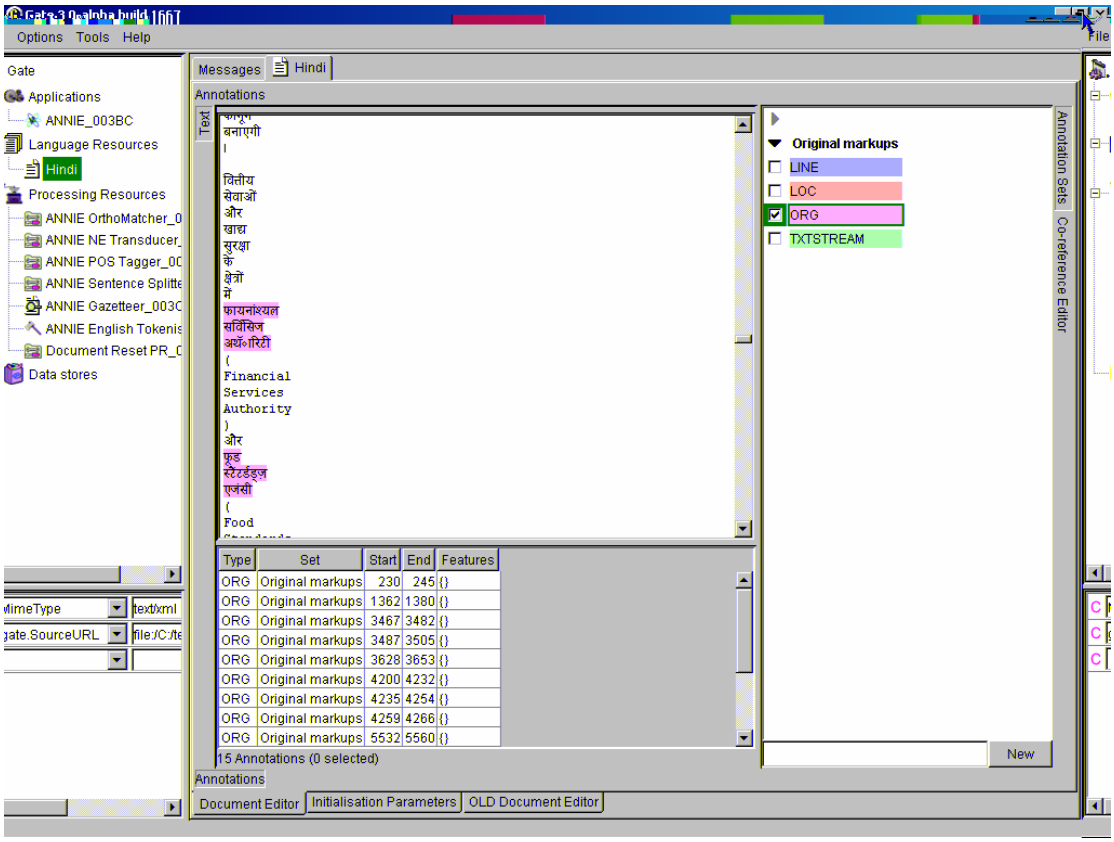

Figure 1 Overview of the GATE GUI

Collectively, the set of resources integrated with GATE is known as CREOLE: a Collection of REusable Objects for Language Engineering. All the resources are packaged as Java Archive (or 'JAR') files, plus some XML configuration data. The JAR and XML files are made available to GATE by putting them on a web server, or simply placing them in the local file space.

In GATE the linguistic information is stored as **annotations** over the original text. The design of the GATE documents is based on the TIPSTER Architecture Design document. Since the version 2, GATE has a similar model, although annotations are now graphs, and instead of multiple spans per annotation each annotation now has a single start/end node pair. The current model is largely compatible with [Bird & Liberman 99], and roughly isomorphic with "stand-off markup" as latterly adopted by the SGML/XML community.

2 GATE as a LIRICS Service Platform

In the context of LIRICS, GATE will be used as the service platform. This means that it will be used for:

- Connecting to LIRICS compliant web services (such as the reference implementations, see deliverable 5.1.E), which means treating them as *ProcessingResources*
- Represent the linguistic information provided by the LIRICS services using the GATE annotation and document format and display the linguistic content using GATE *VisualResources*
- Combine the LIRICS services inside a GATE Application

- Debug the services by displaying the error messages possibly returned by the servers

The LIRICS Platform will be delivered as a CREOLE library of resources which will be loadable as a GATE plugin. Please refer to the GATE documentation for information about how to load a plugin in GATE.

An important distinction is made in GATE between Processing Resources (PR) and Language Resources (LR). A Processing Resource operates on a document and modifies its content (i-e its Annotations), PR's are combined to create an application. At the opposite a Language Resource does not operate on documents (which like corpora, are also LR) and can't modify their annotations. There are different types of Language Resources in GATE, such as ontologies or the WordNet plugin. These resources can be used by a specific PR's to create annotations on documents. This is the case for instance for ontologies, the instances of which can extracted from the texts by a set of GATE PR's.

3 Processing resources

3.1 MAF client

The GATE will contain a new Processing Resource for annotating a document with a MAF Service, such as the reference implementation (see deliverable 5.1.E) which is also based on GATE resources.

As any Processing Resource in GATE, the MAF PR will take as input a GATE document, send its content to the MAFService and convert the XML representation of the MAFDocument into a GATE document and return it. The annotated GATE document might then be processed by another LIRICS Resource, such as the SynAF PR or any other GATE processing Resource, for instance for extracting Named Entities (references to peoples, places, etc…)

The MAF PR will be initialized with a parameter indicating the URL of the MAFService. A runtime parameter will be used to specify the language code of the documents.

The GATE MAF document contains the original textual content and Token and WordForm annotations corresponding to Sections 6 and 7 of the MAF specification (currently ISO TC 37/SC 4 N119 Rev. 3).

Figure 2 shows an example of a document annotated with MAF and displayed as a GATE document.

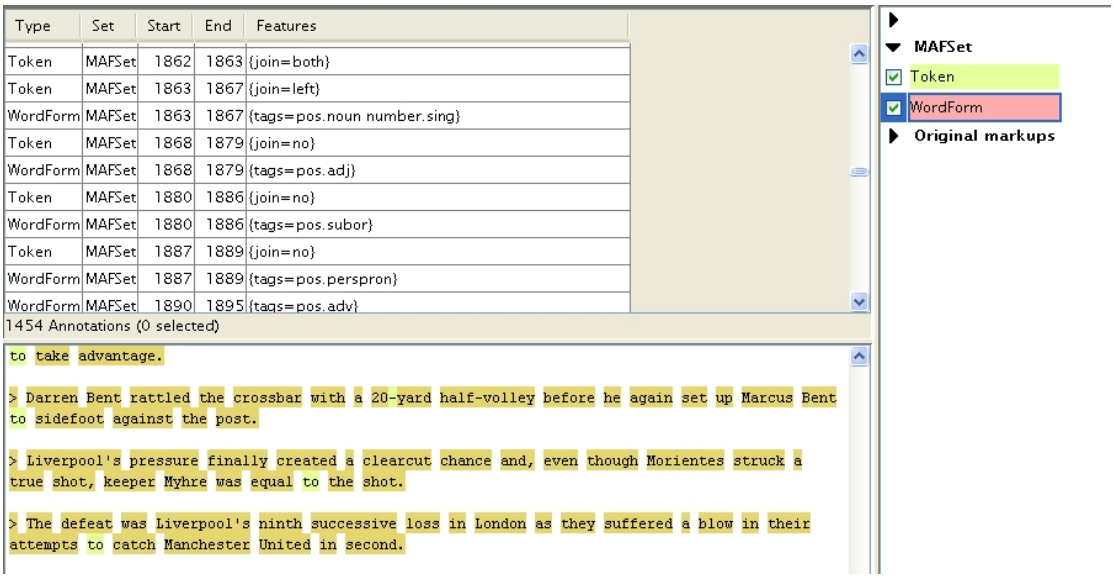

Figure 2 A MAF annotated document in GATE

3.2 SynAF client

A client for the Syntactic Annotation Framework will be provided in version 2 of the integration platform. The API for the SynAF service is described in deliverable D5.1.D.

3.3 Semantic API client

The Service Platform will not contain a Semantic API client, as there will not be any reference implementation for it. Its representation format being similar to the Synaf one, the SynAF client should be able to consume and display a document annotated with regard to the SemAF.

4 Language resources

4.1 LMF client

The LMF reference implementation provides a way to query a lexicon via a web service. Its functionality is to manage, query and retrieve the lexical information stored on a remote server.

The main difference between the LMF client in the Service Platform and the other clients (MAF – SynAF) is that the function of LMF is not necessarily to produce annotations on documents, since the LMF API is more generic than that. As such the most straightforward and useful way to illustrate the use of LMF from the service platform is to implement a client as a Language Resource. This resource would be used to display, query and browse the content of a LMF compliant lexicon.

A similar functionality is already covered in GATE with the Ontology viewer, or closer to LMF, the WordNet plugin which is used to browse the content of Wordnet. The Figure 3 shows a screenshot of this plugin.

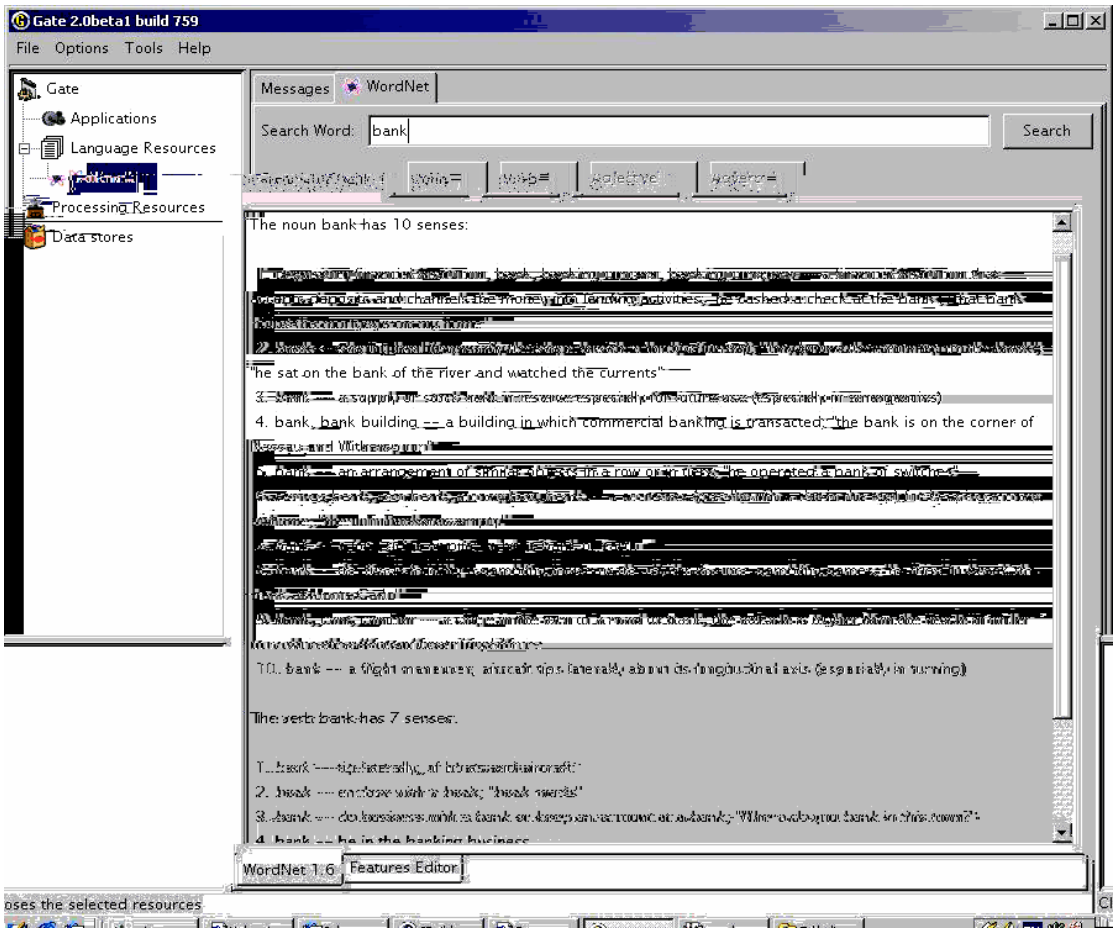

Figure 3 WordNet plugin in GATE

A set of Processing Resources could be created in order to generate annotations in documents using the lexical information stored in a LMF service. However this functionality is not covered in the reference platform as it is too specific in comparison with the scope of the LMF API. It is also meant to be covered by the MAF service, which may generate a lexical information to the WordForm entities.