eContent

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	M30
Actual date of delivery	24 August 2007
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		
2.0	24/07/07		

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.

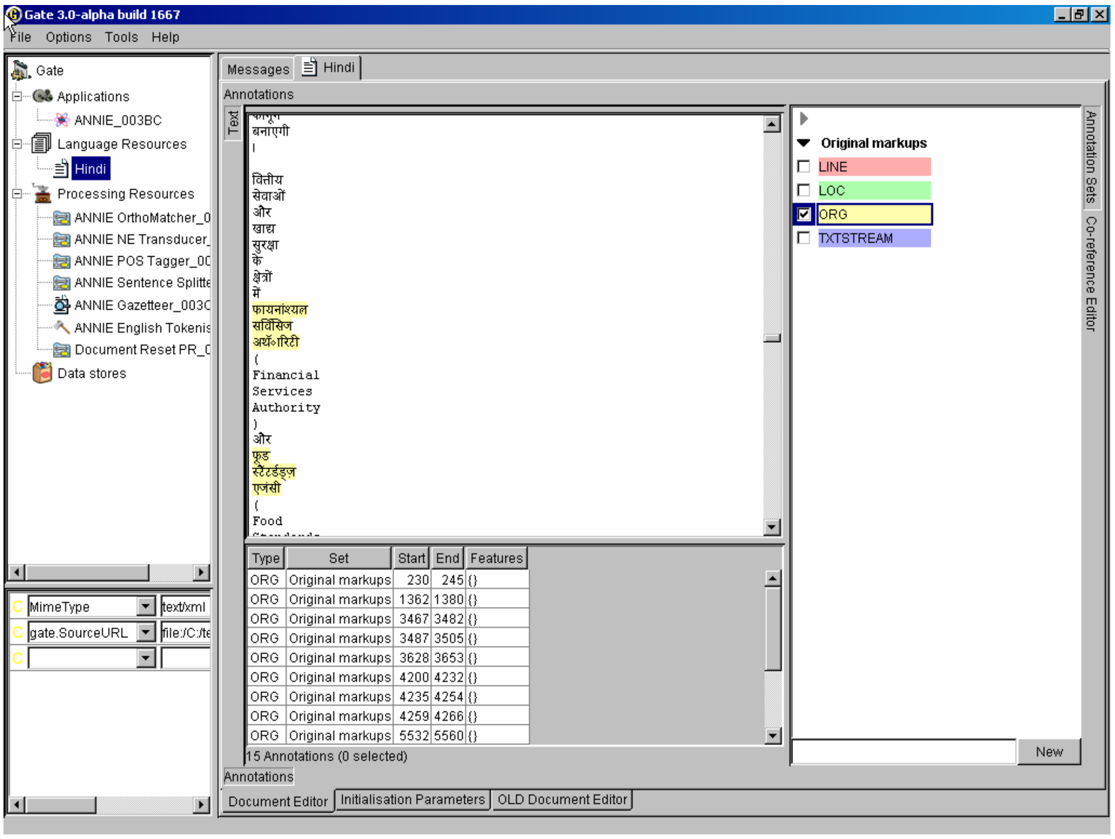
GATE can be defined as:

- 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.



```
बनाएगी
।

वित्तीय
सेवाओं
और
खाद्य
सुरक्षा
के
क्षेत्रों
में
फायनांश्यल
सर्विसिज
अथॉरिटी
(
Financial
Services
Authority
)
और
फूड
स्टैंटर्डइज़
एजंसी
(
Food
```

Type	Set	Start	End	Features
ORG	Original markups	230	245	{}
ORG	Original markups	1362	1380	{}
ORG	Original markups	3467	3482	{}
ORG	Original markups	3487	3505	{}
ORG	Original markups	3628	3653	{}
ORG	Original markups	4200	4232	{}
ORG	Original markups	4235	4254	{}
ORG	Original markups	4259	4266	{}
ORG	Original markups	5532	5560	{}

15 Annotations (0 selected)

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 act as the service platform, used for the following purposes:

- connecting to LIRICS compliant web services (such as the reference implementations described in deliverable 5.1.E), which means treating them as *ProcessingResources;*
- representing the linguistic information provided by the LIRICS services using the GATE annotation and document format and displaying the linguistic content using GATE *VisualResources;*
- combining LIRICS services and other GATE resources inside a GATE Application; and

- debugging the services by displaying the error messages possibly returned by the servers.

The LIRICS Platform will be delivered as a set of CREOLE plugins of resources along with sample GATE applications. Please refer to the GATE documentation for information about loading plugins and creating and using GATE resources.

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 annotations (and possibly, but not usually, its textual content); PRs are combined to create an application. A Language Resource, on the other hand, does not operate on or modify documents and corpora (which are also specific types of LRs). There are different types of LRs in GATE, including documens, corpora (of documents), ontologies and the WordNet interface. LRs can supply data to specific PRs which in turn create and modify annotations on documents but do not directly affect annotations. PRs can also modify the data contained in some LRs (by adding instances to an ontology, for example).

3 Processing resources

3.1 MAF client

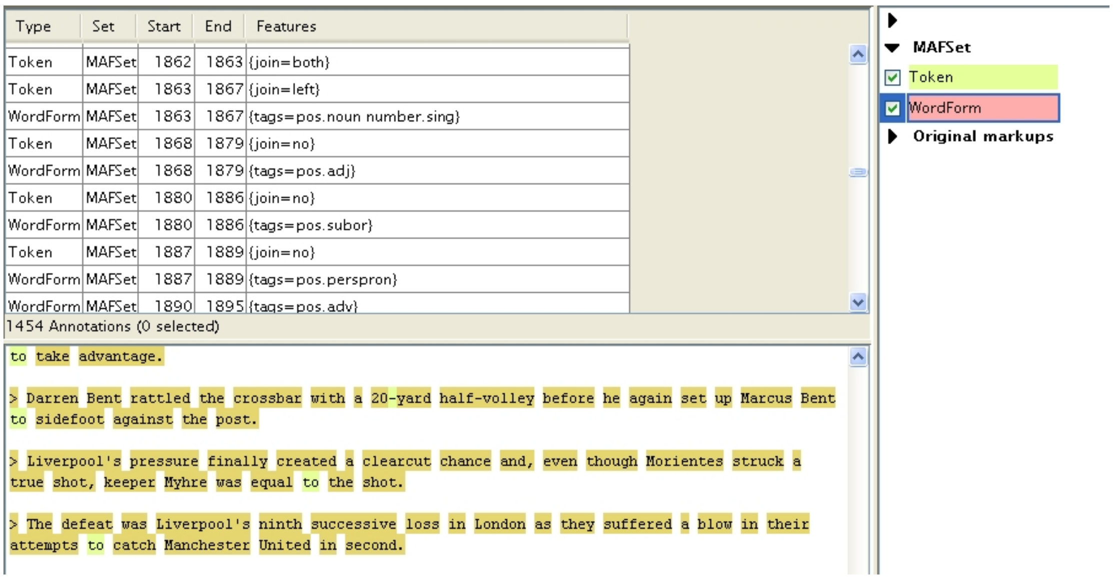
We provide a GATE Processing Resource for annotating a document according to a MAF Service, such as the reference implementation in deliverable 5.1.E (which is also based on GATE resources).

The MAF client PR will take a GATE corpus as input and send the content of each document to the MAF Service, which will return an XML representation of the MAF-annotated document, which the client will convert back into annotations on the local GATE document. The modified GATE document can then be processed by any other GATE PR, such as the SynAF client (see below), a named entity extraction PR or a JAPE transducer.

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. (All the documents in a GATE corpus processed by the MAF client at one time should be in the same language.)

The GATE MAF document contains the original textual content and token and wordForm annotations corresponding to Sections 6 and 7 of the MAF specification. The MAF plugin also extends the GATE document VR so that the document can be saved to disk as MAF XML files from the GATE GUI; this operation also displays the MAF XML in an additional tab of the document window.

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



Type	Set	Start	End	Features
Token	MAFSet	1862	1863	{join=both}
Token	MAFSet	1863	1867	{join=left}
WordForm	MAFSet	1863	1867	{tags=pos.noun number.sing}
Token	MAFSet	1868	1879	{join=no}
WordForm	MAFSet	1868	1879	{tags=pos.adj}
Token	MAFSet	1880	1886	{join=no}
WordForm	MAFSet	1880	1886	{tags=pos.subor}
Token	MAFSet	1887	1889	{join=no}
WordForm	MAFSet	1887	1889	{tags=pos.perspron}
WordForm	MAFSet	1890	1895	{tags=pos.adv}

1454 Annotations (0 selected)

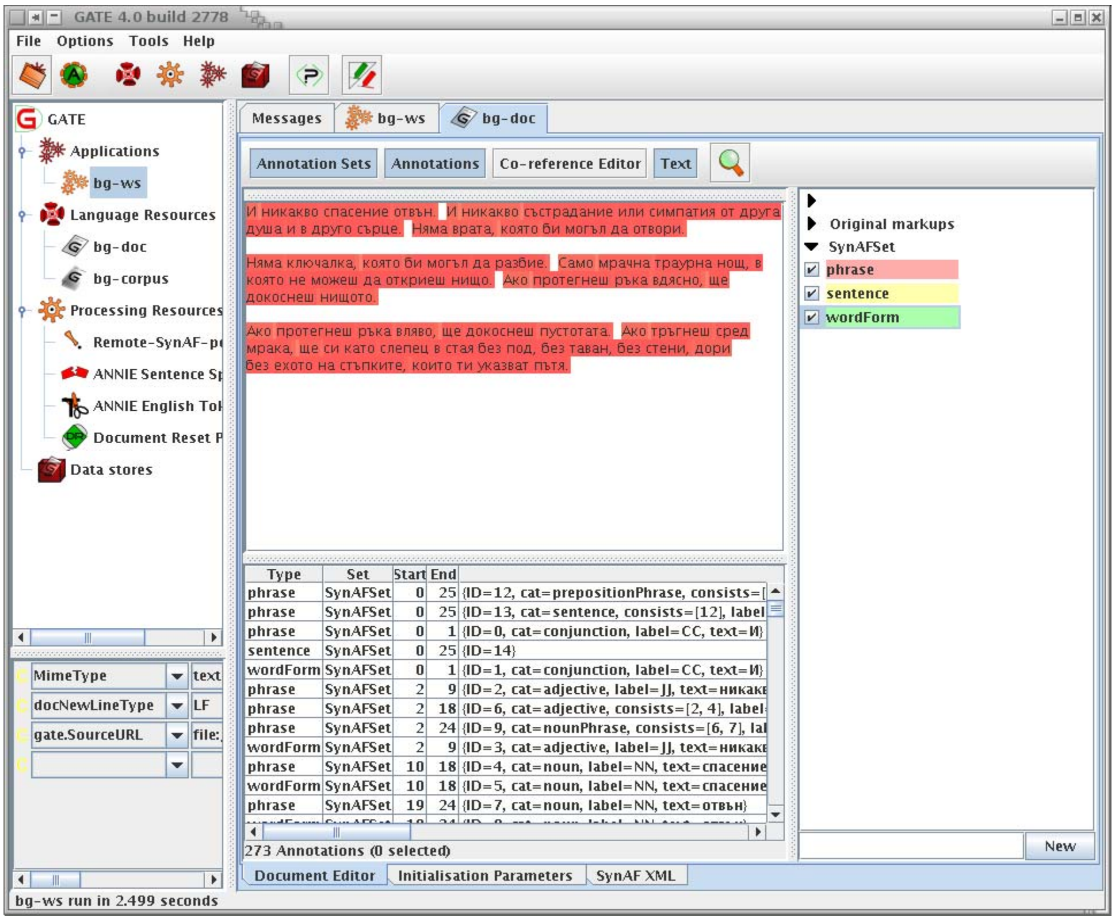
```
to take advantage.

> Darren Bent rattled the crossbar with a 20-yard half-volley before he again set up Marcus Bent
to sidefoot against the post.

> Liverpool's pressure finally created a clearcut chance and, even though Morientes struck a
true shot, keeper Myhre was equal to the shot.

> The defeat was Liverpool's ninth successive loss in London as they suffered a blow in their
attempts to catch Manchester United in second.
```

Figure 2: A MAF annotated document in GATE

3.2 SynAF client

We provide a similar SynAF client PR, which sends the content of each document in a corpus to a SynAF service, which returns an XML representation of the SynAF annotations, which the client transforms back into annotations (of types *sentence*, *phrase*, *relation* and *wordForm*) on the local GATE documents. Figure 3 shows a document in the GATE GUI after SynAF processing.

As with the MAF client, the documents can be subsequently processed by other PRs and saved to disk as SynAF XML files. In addition to the document tab which displays the SynAF XML after "Save as SynAF XML" has been carried out (shown in Figure 4), the plugin provides a syntax tree viewer which displays the constituency relationships; to use this, right-click on a sentence annotation in the annotations pane and select "Edit with SynAF Tree Viewer". Figure 5 shows such a tree.



И никакво спасение отвън. И никакво състрадание или симпатия от друга душа и в друго сърце. Няма врата, която би могъл да отвори.

Няма ключалка, която би могъл да разбие. Само мрачна траурна нощ, в която не можеш да откриеш нищо. Ако протегнеш ръка вдясно, ще докоснеш нищото.

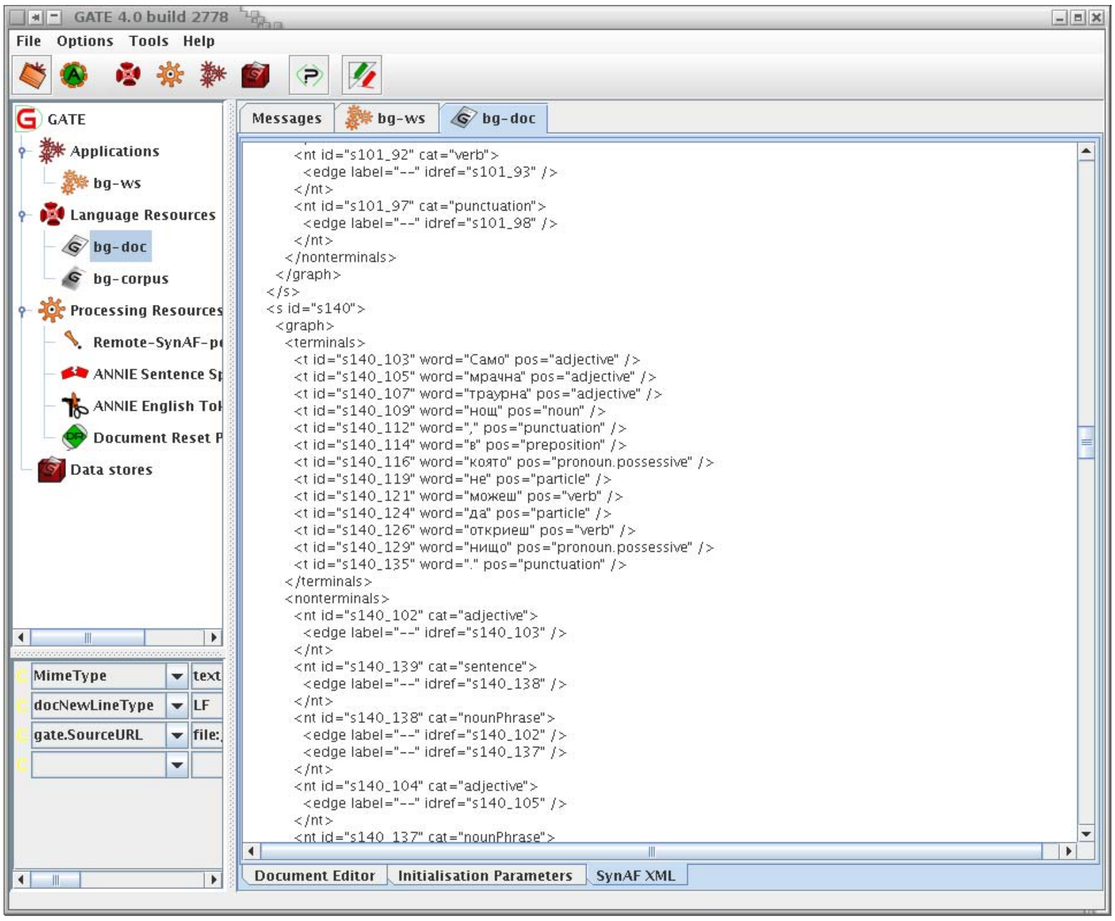
Ако протегнеш ръка вляво, ще докоснеш пустотата. Ако тръгнеш сред мрака, ще си като слепец в стая без под, без таван, без стени, дори без ехото на стъпките, които ти указват пътя.

Type	Set	Start	End	
phrase	SynAFSet	0	25	{ID=12, cat=prepositionPhrase, consists=[
phrase	SynAFSet	0	25	{ID=13, cat=sentence, consists=[12], label
phrase	SynAFSet	0	1	{ID=0, cat=conjunction, label=CC, text=И}
sentence	SynAFSet	0	25	{ID=14}
wordForm	SynAFSet	0	1	{ID=1, cat=conjunction, label=CC, text=И}
phrase	SynAFSet	2	9	{ID=2, cat=adjective, label=JJ, text=никакв
phrase	SynAFSet	2	18	{ID=6, cat=adjective, consists=[2, 4], label
phrase	SynAFSet	2	24	{ID=9, cat=nounPhrase, consists=[6, 7], la
wordForm	SynAFSet	2	9	{ID=3, cat=adjective, label=JJ, text=никакв
phrase	SynAFSet	10	18	{ID=4, cat=noun, label=NN, text=спасение
wordForm	SynAFSet	10	18	{ID=5, cat=noun, label=NN, text=спасение
phrase	SynAFSet	19	24	{ID=7, cat=noun, label=NN, text=отвън}

273 Annotations (0 selected)

bg-ws run in 2.499 seconds

Figure 3: A SynF annotated document in GATE



```
      <nt id="s101_92" cat="verb">
        <edge label="--" idref="s101_93" />
      </nt>
      <nt id="s101_97" cat="punctuation">
        <edge label="--" idref="s101_98" />
      </nt>
    </nonterminals>
  </graph>
</s>
<s id="s140">
  <graph>
    <terminals>
      <t id="s140_103" word="Само" pos="adjective" />
      <t id="s140_105" word="мрачна" pos="adjective" />
      <t id="s140_107" word="траурна" pos="adjective" />
      <t id="s140_109" word="нощ" pos="noun" />
      <t id="s140_112" word="," pos="punctuation" />
      <t id="s140_114" word="в" pos="preposition" />
      <t id="s140_116" word="която" pos="pronoun.possessive" />
      <t id="s140_119" word="не" pos="particle" />
      <t id="s140_121" word="можеш" pos="verb" />
      <t id="s140_124" word="да" pos="particle" />
      <t id="s140_126" word="откриеш" pos="verb" />
      <t id="s140_129" word="нищо" pos="pronoun.possessive" />
      <t id="s140_135" word="." pos="punctuation" />
    </terminals>
    <nonterminals>
      <nt id="s140_102" cat="adjective">
        <edge label="--" idref="s140_103" />
      </nt>
      <nt id="s140_139" cat="sentence">
        <edge label="--" idref="s140_138" />
      </nt>
      <nt id="s140_138" cat="nounPhrase">
        <edge label="--" idref="s140_102" />
        <edge label="--" idref="s140_137" />
      </nt>
      <nt id="s140_104" cat="adjective">
        <edge label="--" idref="s140_105" />
      </nt>
      <nt id="s140_137" cat="nounPhrase">
```

Figure 4: SynAF XML displayed in GATE

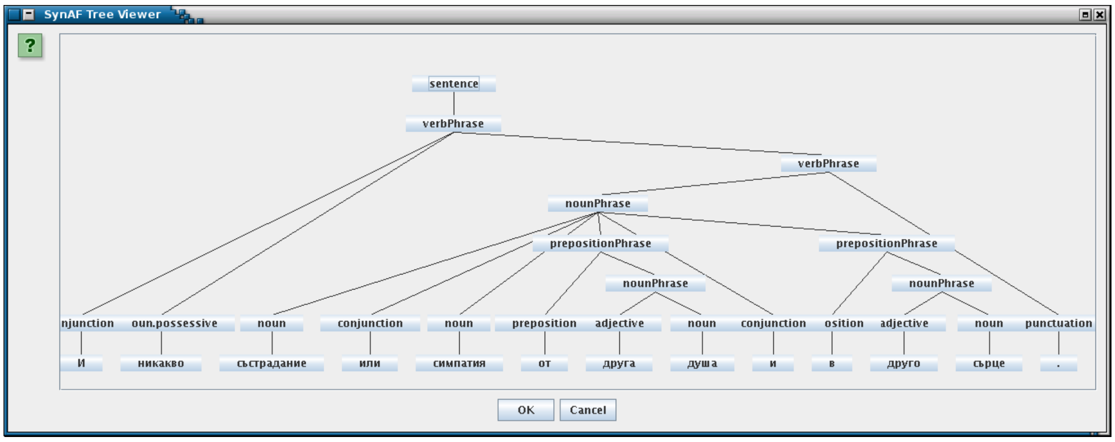


Figure 5: A SynAF constituency tree displayed in GATE

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 can used to display, query and browse the content of a LMF compliant lexicon.

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 lexical information in the wordForm entities.