# Uniform Language Resource Access and Distribution in the Context of GATE – a General Architecture for Text Engineering

Hamish Cunningham, Kalina Bontcheva, WimPeters, Yorick Wilks

Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello Street
Sheffield, S1 4DP, UK
{hamish, kalina, wim, yorick}@dcs.shef.ac.uk

**Abstract.** The reuseability and accessibility of lexical and ontological resources often requires substantial programming overhead and detailed knowledge of the structure and nature of resource-specific information. Each resource has its own representation syntax and covers a particular subset of linguistic phenomena.
This paper will discuss ways to overcome these barriers to resource reuse and presents a new method for distributing and accessing language resources within the GATE environment (General Architecture for Text Engineering). In this environment linguistic resources are modelled in an object-oriented way, yielding an initial set of resource-specific inheritance hierarchies. The exploitation of commonalities between resources leads to an incremental integration of their hierarchies into one common object model (whilst preserving the originals). GATE also offers the necessary mechanisms for storage, access and editing of lexical resources which can reside both on local or remote computers.

## 1 Introduction and Background

In general, the reuse of NLP data resources (such as lexicons or corpora) has exceeded that of algorithmic resources (such as lemmatisers or parsers) [CFB94]. Although data for specific tasks often needs to be created from scratch or extracted from existing resources, nevertheless, for English at least, there exists a sizeable number of resources that are reusable and whose contents can serve multiple purposes. However, there are still two barriers to data resource reuse:

1. each resource has its own representation syntax and corresponding programmatic access mode (e.g. SQL for Celex, C or Prolog for WordNet);
2. resources must generally be installed locally to be usable, and how this is done depends on what operating systems are supported etc., which varies from site to site.

A consequence of (1) is that although resources of the same type usually have some structure in common (for example, at one of the most general levels of description lexicons are organised around words), this commonality cannot be exploited when it comes to using a new resource. In each case the user has to adapt to a new data structure which brings significant overhead. Work which seeks to investigate or exploit commonalities between resources has first to build a layer of access routines on top of each resource. So, for example, if we wish to do task-based evaluation of lexicons, by measuring the relative performance of an information extraction system with different instantiations of lexical resource, we might have to write code to translate several different resources into SQL or some other common format.

A consequence of (2) is that there is no way to "try before you buy": no way to examine a data resource for its suitability for one's needs before licencing it. Correspondingly there is no way for a resource provider to give limited access to their products for advertising purposes, or gain revenue through piecemeal supply of sections of a resource.

This paper will discuss ways to overcome these barriers. We will present a new method for distributing and accessing language resources involving the development of a common programmatic model of the various resource types, implemented in UML (Unified Modelling Language, the new lingua franca of object-oriented modelling [Fow97]) and Java, along with a framework for distributed storage and access of these resources. This model and server are being designed as part of GATE (General Architecture for Text Engineering [CHGW97]) and go under the provisional title of an Active CREOLE Server. (CREOLE: Collection of REusable Objects for Language Engineering).

## 2 Objectives

Our aim is to create a common object-oriented model for language data resources that encapsulates the union of the linguistic information contained in a range of resources, and will encompass as many object hierarchies as there are resources. At the top of the resource hierarchies are very general abstractions (e.g. the 'head concepts' in any thesaurus like WordNet); at the leaves are data items specific to individual resources. Programmatic access is available at all levels, allowing the developer to select an appropriate level of commonality for each application. Generalisations will be made over different object types in the resources, and the object hierarchies will be merged at whatever levels of description are appropriate. No single view of the data will be imposed on the user, who may chose to stay with the "original" representation of a particular resource, or to access a model of the commonalities between several resources, or a combination of both: our aim is above all inclusive, and this is *not* a new standards initiative, but a way to build on previous initiatives ("standardised" and otherwise).

The object-oriented architecture will represent the linguistic data in a conceptual way. The user will be able to think about a word and its properties without having to assess the actual structure of the data storage. This is in contrast to other data structures such as the relational format where the information about objects is often scattered over many relations or records [EN94] (p.665), which obscures the conceptual transparency of the database. It is also more efficient than textual representation in e.g. SGML, which only handles trees fluently, and does not support random access.

A common object model, sitting on top of the resource-specific models, will allow a uniform access procedure for all resources. This will reduce the amount of programmatic overhead referred to above, and be beneficial to the user in other ways. The object-oriented model will allow the user to:

- select the information needed in a conceptual manner, without first having to select the most appropriate resources by detailed examination and comparison, or to extract the information needed in a way determined only by the original structure of the resource in question;
- identify overlaps and differences between the resources;
- assess the usability of each resource for specific language processing tasks.

The users of linguistic data can be divided roughly into two camps: linguists and computer scientists. Conceptual unification of linguistic data structures will help both in different ways. The former will no longer be confused by having to reassess each resource and dedicate time to computational tasks which should in principle be unnecessary for their work. The latter are often lacking linguistic knowledge, and will benefit because they will only have to interpret one kind of data structure.

In general, an object-oriented design will enhance the conceptualisation, reusability and integratability of the resources for all types of users.

However, the user can only benefit fully from such an uniform, object-oriented model of language resources only if it comes together with tools that allow easy storage, manipulation, and visualisation of such data. This is particularly true if the user is merging several existing resources using the model and needs to store and access the new one in the future. In addition, distributed resource access requires support for efficient and secure communication between the resource server and the client application. Finally, the user might wish to work with both local and remote resources in a uniform, transparent way without having to learn about TCP/IP sockets, Java RMI, etc.

To summarise, we are developing three things:

1. an OO model of each resource, documented in UML and accessible in Java, that sticks very closely to the structures and terminology of the resource as it currently exists (e.g. for WordNet, accessor methods on our objects will mirror the functionality of the existing C and Prolog APIs);
2. a unifying model layered on top of the resource-specific models, that captures generalisations about them (implemented in the same way);
3. a data management substrate that handles distributed storage, data visualisation and efficient access to the data in each resource modelled in accordance with points 1 and 2.

Now we will continue with a brief overview of GATE, following by an in-depth discussion on point 3. Some issues related to points 1 and 2 will be also covered briefly but for an in-depth discussion see [PCM+98].

## 3 GATE - a General Architecture for Text Engineering

The rationale behind the existance of GATE is that researchers in language processing, ontologies, and similar areas should not need to care about each of the following problems, each time they are working on a new system or language resource:

- store data on disk;
- display data;
- load processor modules and data stores into processes;
- initiate and administer processes;
- divide computation between client and server;
- pass data and control between processes and machines.

A *Software Architecture* for language processing should do all this for you, just like systems like Mathematica and SPSS support researchers in other fields in their daily tasks.

GATE (General Architecture for Text Engineering)[1] [CHGW97,CGHW99] is one such architecture that aims at providing such much needed tools. In a nutshell, it is a framework providing representation and storage of language data together with infrastructural support for building and deploying language engineering applications. Its first version offers plug-and-play support for processing modules which is now extended to support re-use of large-scale language resources (e.g., ontologies, lexical knowledge bases) and customisable data visualisation modules.

Version 1 comprises three principal elements (see figure 1):

- GDM, the GATE Document Manager, based on the TIPSTER document manager [Gri97];
- CREOLE, a Collection of REusable Objects for Language Engineering: a set of LE components integrated with the system;
- GGI, the GATE Graphical Interface, a development tool for LE R&D, providing integrated access to the services of the other components and adding visualisation and debugging tools.

Following user feedback on the interface usability, GATE's GGI was improved in GATE 2 to provide a look-and-feel that follows de-facto standards in modern direct manipulation interfaces, in order to reduce the learning overhead. The new development environment shows all loaded language resources (e.g., documents), available processing resources and systems created by the user (see Figure 2).

The interface provides tabbed views, direct manipulation of GATE entities, drag and drop, flexible customisation (e.g., which annotations are highlighted in the text, which viewers can display annotations of a given type). In addition,

[1] GATE documentation, publications, demos are available on `http://gate.ac.uk`. For download and install instructions for GATE v.1.5 follow the link to version 1 from the GATE site.

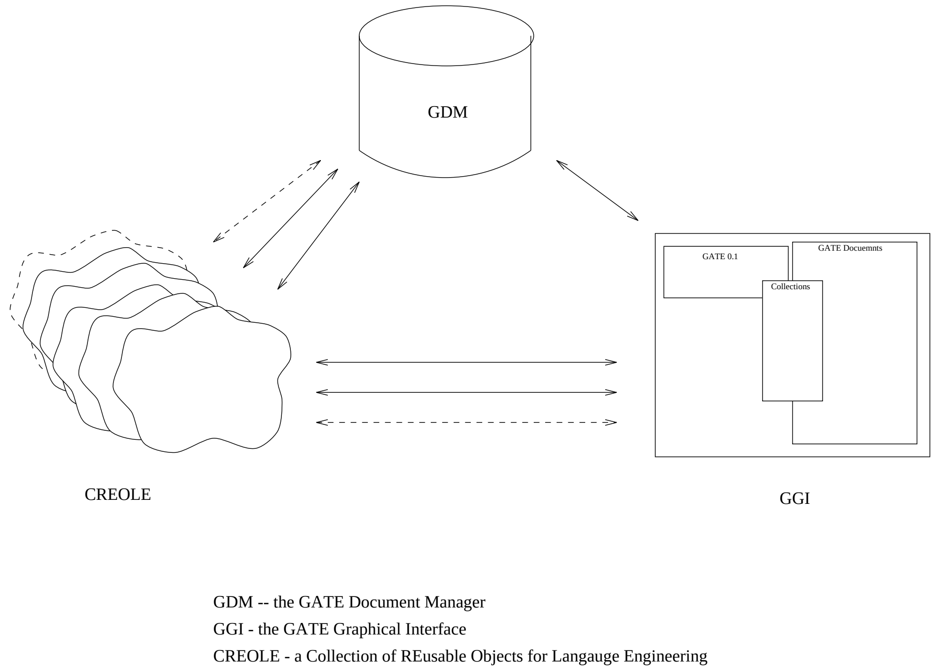


**Fig. 1.** The three principal elements

all viewers (e.g., the collection viewer shown in the right-hand part of Figure 2) can be reused and the interface can be extended with user-supplied ones. The viewer independence from the development environment also enables their deployment as parts of end-user applications (e.g., a Netscape plugin that detects and displays named entities in e-mail and Web pages).

As discussed in [Cun00], the evaluation and user feedback on GATE 1 also underlined the need for:

- efficient storage, access, and visualisation of language resources;
- easy integration and re-use of existing lanaguage resources (e.g., WordNet, LDOCE);
- support for distributed applications which can access such resources over the net.

The following section will discuss how GATE version 2 addresses each of these issues.

## 4 Distribution and Access of Language Resources

In addition to the corpora and documents in GATE 1, support for Language Resources (LRs) such as lexicons, ontologies and thesauri has now been added. All these resources are modelled as data source objects (in general these are

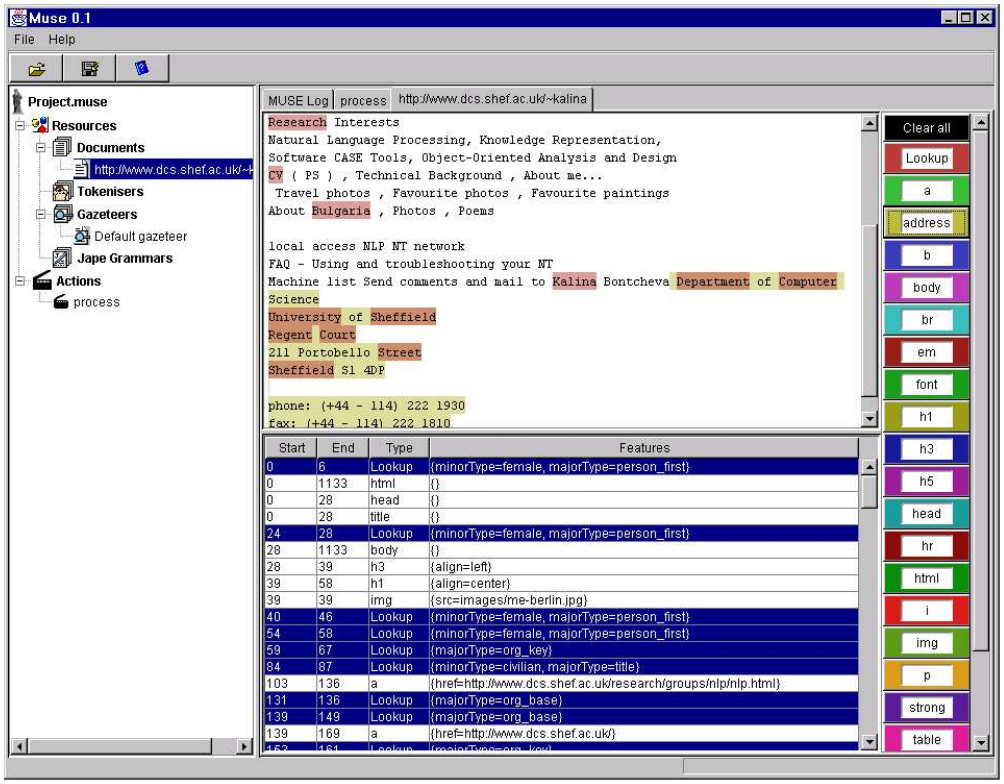

**Fig. 2.** A customised GATE 2 development environment

implemented using relational databases) coupled with a Java abstraction layer that encapsulate their structure in an OO form. Taking advantage of the Java DataBase Connectivity (JDBC) standard, the architecture allows the set of LRs used by an application to be distributed across the Net. In this way, the common Java interface isolates the user from tedious low-level database operations such as establishing connections, authentication, and transaction management.

## 5 LR access

GATE now provides uniform, simple methods for accessing language resources regardless of the way data is stored (e.g. files vs. relational databases) and retrieved (e.g. file seek vs SQL query). For example, an instance of WordNet may be created by constructing a `DataStore` object from a reference to the URL of a database that stores the thesaurus data (the system communicates with the database using the JDBC protocol, which provides both distribution and vendor independence). The client then asks the `DataStore` for a WordNet object, which is returned as a `Thesaurus` interface; this interface is used to obtain the necessary data from the resource (it may also make SQL queries on the object if required). If the client has permission to update the WordNet data, they can call mutator methods on the interface, then ask the `DataStore` to make the changes permanent, or discard them as necessary.

### 5.1 Integrating LRs in GATE

The GATE framework provides support for bootstrapping, discovery, loading, and access to Language Resources (LRs), as part of its general mechanism for plug-and-play support of data, processing, and visualisation components[2].

*Resource bootstrapping* helps the developer with the task of integrating a resource in the system by providing some automatically generated stub code which represents an empty resource. The developer then adds some code that implements the resource or creates a wrapper code that mediates the communication between GATE and an existing resource (e.g. WordNet). At start-up GATE uses the resource description files generated during bootstrapping to *discover* and *load* all available resources. These resources then become available to be used in the process of building and testing language processing systems inside GATE.

### 5.2 Viewing and Editing

In order to support the user with common LR manipulation tasks, such as data viewing and editing, GATE offers some specialised graphical tools. Each LR type is associated with one or more visualisation components. For example, there are components for displaying trees, single-range structures such as part-of-speech tags, and multi-range structures such as coreference chains. Where appropriate, the display components also allow editing (see for example the syntax tree viewer/editor at http://gate.ac.uk/demos/).

LR-specific viewing/editing tools (e.g., an ontology editor, WordNet data viewer) can be easily integrated in GATE as long as they support the JavaBeans standard.

## 6 Capturing Resource Commonalities

In order to bootstrap our work on the resource-specific and the unified resource models discussed in Section 2, we have begun work on analysing the following six resources: WordNet [ME90], Comlex [MG94], Celex [Bur90], EuroWordNet [VDOP97], CRL-LDB [WGS96] and Mikrokosmos [OBN96]. They are available from the owners or distribution organisations such as ELRA and the Linguistic Data Consortium. We will not in general seek to redistribute them, but to provide the code that maps them into our object model in such a way as to enable others in the community to use their own separately licenced copies with the model and server.

[2] *Resource*, *component* and *CREOLE object* are all synonyms and are used interchangeably.

These resources vary in both linguistic scope and granularity, and each resource covers one or more linguistic areas. Comlex contains mostly orthographic and morpho-syntactic information; Celex mainly has orthographic, morphological, syntactic and phonological data, whereas WordNet, EuroWordNet and MikroKosmos provide mostly semantic information. As table 1 shows via a comparison of a number of linguistic features in four of the six resources, not all of them have the same coverage or granularity of description and none of them share the same linguistic conventions to describe (potentially) the same features.

| | CRL-LDB | CELEX | COMLEX | WordNet |
|---|---|---|---|---|
| **Orthography:** | | | | |
| spelling variants | Y | Y | Y | Y |
| syllabification | Y | Y | | |
| **Morphosyntax:** | | | | |
| part of speech | Y | Y | Y | Y |
| grammatical subcategory | Y | Y | Y | |
| inflectional characteristics | Y | Y | Y | Y |
| **Morphology:** | | | | |
| derivational information | | Y | | |
| **Phonology:** | | | | |
| pronunciation | Y | Y | | |
| **Syntax:** | | | | |
| verb subcategorisation | Y | Y | Y | Y |
| **Semantics:** | | | | |
| sense distinction | Y | | | Y |
| verb argument preferences | Y | | | Y |
| ontological classification | Y | | Y | Y |
| semantic relations | | | | Y |
| domain | Y | | | |

**Table 1.** Linguistic features contained in four resources

As we have previously discussed in more detail [PCM+98], our approach is to both:

1. model the native structure of the resources separately and
2. integrate the resources at various levels of description

For task 1, we model exactly the data and the conventions (or terminology) in each resource. The modelling is guided by the existing data structures and whatever documentation is available. For instance, Celex provides several types of phonetic transcription for lemmata and wordforms in a relational structure. These types result in different phonetic transcription objects: plain, syllabified and syllabified with stress. Since we are modelling only resource-specific choices at this stage, we maintain all of this information at this level of granularity and the conventions that are used to encode it. We follow the same procedure for each resource.

For task 2, we compare, extend and incrementally combine the resource-specific models, thus creating a normalisation of resource-specific information into one common object model. However, by continuing to maintain the base, "pure", models of the individual resources, the user always has the option to use the resource-specific format they are familiar with.

Pooling these resources together into one resource can yield a richer common data structure than any single resource can provide. Our common model will maximise access to the total range and variety of data whilst maintaining the individual distinctions between each resource and combining them in a linguistically-controlled environment.

In this context, the issue of standards is a vexed one: experience with repositories of lexical materials (e.g. the CRL Consortium for Lexical Research 1989-93) suggested that if resources had to have standardised formats, they would not be deposited or used. The success of WordNet worldwide is a demonstration of how researcher choice can defy any committee's standards. What we propose here is quite different from projects like SEAL [EK95] that attempt to conflate different lexical resources: in what we propose, the resources retain their integrity, or "native" structure. We propose, via an object oriented methodology, a standardised taxonomy and structure only as an index to the links between lexical objects with the same function in the various resources. Those wishing to research the merging of resources will be one of the beneficiaries of our work, but this is not our primary aim.

## 7 Conclusion

In this paper we argued for the need of an uniform framework for storage, access, and manipulation of language resources. Here we also described briefly GATE - a General Architecture for Text Engineering and the ways in which it supports these tasks by offering:

1. an OO model of each resource that follows closely the structures and terminology of the resource as it currently exists;
2. a unifying model layered on top of the resource-specific models, that captures generalisations about them;
3. a data management substrate that handles distributed storage, data visualisation and efficient access to the data in each resource modelled in 1 and 2.

An important benefit of this work will be the creation of an appropriate environment for the evaluation of resources. Whereas at present it is practically very difficult to treat different resources in the same way for purposes of evaluation, the new access method we propose will make this much more feasible, and provide a firm basis for comprehensive evaluation efforts.

The plug-and-play nature of GATE will also allow developers of ontologies and lexical knowledge bases to integrate their own resources, so they can be used together with already supported ones (e.g., document corpora). By using GATE for their resource development environment users will also benefit from available viewing and editing tools, which they can also extend for new types of data. Support for all this functionality is currently under development; for news about the forthcoming GATE version 2 release keep checking `http://gate.ac.uk`.

## References


[Bur90] G. Burnage. *CELEX, A Guide for Users.* CELEX Centre for Lexical Information, Nijmegen, The Netherlands, 1990.

[CFB94] H. Cunningham, M. Freeman, and W.J. Black. Software Reuse, Object-Oriented Frameworks and Natural Language Processing. In *Proceedings of the conference on New Methods in Natural Language Processing (NeMLaP-1)*, Manchester, 1994.

[CGHW99] H. Cunningham, R.G. Gaizauskas, K. Humphreys, and Y. Wilks. Experience with a Language Engineering Architecture: Three Years of GATE. In *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Edinburgh, April 1999. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.

[CHGW97] H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. Software Infrastructure for Natural Language Processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 237–244, March 1997. Available as `http://xxx.lanl.gov/ps/9702005`.

[Cun00] Hamish Cunningham. *Software Architecture for Language Engineering.* PhD thesis, University of Sheffield, 2000. `http://gate.ac.uk/sale/thesis/`.

[EK95] R. Evans and A. Kilgariff. Mrds, dictionaries, and how to do lexical engineering. In *Proceedings of the 2nd language engineering convention*, pages 125–132. London, UK, 1995.

[EN94] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings, Redwood City, California, 1994.

[Fow97] M. Fowler. *UML Distilled: Applying the Standard Modelling Language.* Addison-Wesley, 1997.

[Gri97] R. Grishman. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA, 1997. Available at `http://www.tipster.org/`.

[ME90] G. A. Miller (Ed.). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[MG94] C. Macleod and R. Grishman. *COMLEX Syntax Reference Manual.* Proteus Project, NYU, 1994.

[OBN96] B. Onyshkevych, Boyan, and S. Nirenburg. Microkosmos. *Machine Translation 10:1-2 (Special Issue on building lexicons for MT)*, 1996.

[PCM+98] W. Peters, H. Cunningham, C. McCauley, K. Bontcheva, and Y. Wilks. Uniform Language Resource Access and Distribution. In *Workshop on Distributing and Accessing Lexical Resources at Conference on Language Resources Evaluation, Granada, Spain*, 1998.

[VDOP97] P. Vossen, P. Diez-Orzas, and W. Peters. The Multilingual Design of EuroWordNet. In *Proceedings of the ACL/EACL-97 workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, Madrid, 1997.

[WGS96] Y. Wilks, L. Guthrie, and B. Slator. *Electric Words.* MIT Press, Cambridge, MA, 1996.