[*MT Summit III*, Washington, DC, USA]

An Architecture Sketch of EUROTRA-II

Jörg Schütz
IAI
Martin Luther Straße 14
D-6600 Saarbrücken
joerg@iai.uni-sb.de

Gregor Thurmair
SNI A.G.
Otto-Hahn-Ring 6
D-8000 München 83
metal@ztivax.siemens.com

Roberto Cencioni
CEC, DGXIII, B/5
Jean Monnet Building
L-2920 Luxembourg
r_cencioni@eurokom.ie

Abstract

This paper outlines a new architecture for a NLP/MT development environment for the EUROTRA project, which will be fully operational in the 1993-94 time frame. The proposed architecture provides a powerful and flexible platform for extensions and enhancements to the existing EUROTRA translation philosophy and the linguistic work done so far, thus allowing the reusability of existing grammatical and lexical resources, while ensuring the suitability of EUROTRA methods and tools for other NLP/MT system developers and researchers.

1 Introduction

EUROTRA is a Community research and development programme for the creation of a machine translation system of advanced design, capable of dealing with all official languages of the Community.

The EUROTRA programme has led to the creation of a research prototype system operational in a narrow subject field and for limited text types, with a vocabulary of approximately 20,000 entries in each language. The underlying software environment has been developed by a number of contractors, working in close cooperation with EUROTRA researchers.

In November 1990 the Council has adopted a specific programme concerning the preparation of the development of an operational EUROTRA system. The Commission is now executing this programme.

Furthermore, within the third Framework Programme (1990-1994), in its proposal for a specific programme concerning Telematic Systems of General Interest, the Commission has included an action line on Linguistic Research and Engineering.

In preparation of these initiatives, the Commission has carried out a number of feasibility and design studies enabling it to define the overall architecture and the underlying linguistic and computational tools of a powerful development environment, suitable for large-scale research application prototyping purposes ([CEC, 1989]). Such a development environment should be used inside as well as outside EUROTRA follow-up programmes as a common platform for machine translation and other applications involving natural language processing components.

In this paper we outline the basic concepts of the EUROTRA-II development environment which are based on the outcome of the requirements definition and feasibility phases of the EUROTRA-6/2 (ET-6/2) design study ([Gajek *et al.,* 1990], [Gajek *et al.,* 199l]) for a new EUROTRA software environment. This study was carried out by a consortium consisting of the Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung (IAI), Saarbrücken, as the Commission's main contractor, and GAP GEMINI SCS BeCom GmbH, Hamburg, and Siemens-Nixdorf Informationssysteme (SNI) A.G., Munich, as IAI's subcontractors, in close collaboration with the Commission's EUROTRA central software team.

As the existing EUROTRA MT system is demonstrated at the conference's exhibition the audience will have the possibility to compare the two system design approaches to each other. The EUROTRA-I prototype system is a robust and comprehensive development system, suitable for research and development operations in a laboratory environment.

The application software, written in Prolog (YAP) and C, is available on a broad range of Unix platforms, and is built around a number of commercial packages (Prolog compiler, relational DBMS, SGML parser, etc.). It has been specifically designed for use with the current rule formalism, the so-called E-framework, on ordinary text terminals connected to multi-user Unix servers. Its main shortcomings are its limited potential for growth and evolution, due to the lack of a uniform and consistent interface between individual components, and the poor run-time performance of the rule formalism.

The new architecture design of the system will ensure the opening up of EUROTRA to other NLP/MT system developers and researchers, but it will also provide a platform for continuity, extensions and enhancements to the existing EUROTRA translation philosophy.

Great emphasis has therefore been placed on the openness and modularity of the system architecture, so that individual components can be developed, enhanced and/or customized by third parties. One of the key design aims is to ensure the reusability of the linguistic resources created with the tools provided by the development environment.

2 The global EUROTRA-II Architecture

2.1 User Profiles

The typical user of the EUROTRA-II environment will be either a skilled researcher in computational linguistics or a team of researchers, who will be provided with a fully developed software environment enabling them to produce large-scale linguistic descriptions of different languages for a number of NLP/MT application domains.

It might be also an application designer or developer who wants to enhance or add other functionalities to the EUROTRA-II environment in order to develop a specific application prototype.

2.2 Requirements and Design Aims

We propose an architecture for the EUROTRA-II software environment which satisfies the following basic requirements:

- The architecture is based on a relatively conservative approach that ensures an efficient implementation on mid-sized Unix workstations.
- It is largely formalism independent and thus reusable for other NLP/MT projects.
- It is modular and (user) configurable.
- It is open to further developments and functional extensions.
- Its design allows for a multi-user NLP/MT development environment.
- It is user-friendly and robust, including error recovery and exception handling capabilities,
- The technical solution adopted for its design and implementation are based on standard state-of-the-art products, e.g. commercially available data base management systems, and techniques, e.g. X-protocol, OSF/Motif widget set, etc., as well as on widely recognized international standards: thus, for instance, the ISO 8859 character set supports the community languages and preserves the multilingual capabilities of the environment,
- The process overhead is as small as possible to gain maximum time and space efficiency.
- Last but not least the system is portable and can be used on different standard, e.g. POSIX and X/Open, Unix platforms.

2.3 Global Architecture

The basic idea of the EUROTRA-II software environment architecture as outline in this paper is derived from research and developments in the fields of distributed artificial intelligence (DAI) and object-oriented programming (OOP).

The EUROTRA-II lingware development environment for NLP/MT applications is modelled through cooperating software agents. Its user environment implements windows and menus with graphic symbols (icons) instead of text commands, to provide an intuitive man-machine interface.

We define a NLP/MT application as a specific instantiation of a problem solving process. For example, if we want to analyse a text, we have to solve several sub-problems (morphology, syntax, semantic, references within and across sentence boundaries, text domain specific knowledge, etc.) in order to achieve a representation of the meaning of the text as a final result of the analysis problem. In our architecture such sub-problems are resolved by specific agents which are embedded in a problem context dependent environment.

The idea, then, is to model a problem solving behaviour of the whole environment, where possible solutions of a given problem ran only be achieved by combining the problem solving capabilities distributed among a number of (potentially highly autonomous) agents, none of which has the ability to solve the whole problem on its own.

The advantage of such an architecture is its open and modular design, which allows the replacement or addition of agents. The design of individual agents caters for changes in their internal problem solving capability. Last but not least, this architecture allows for the distribution of the agents over several machines; in this case only the agent which is responsible for the overall environment supervision is active on the local machine.

The disadvantage of a highly distributed system architecture is, however, the coordinating, monitoring and synchronizing of the control flow within a given application, e.g. interrupt handling, signal propagation, multi-user scheduling etc..

In order to have an architecture that is feasible, efficient and reliable on existing Unix software and hardware platforms, we therefore assume that the agents in the initial (i.e. 1991-1993) implementation of the software environment are invoked and scheduled by one monitoring agent which is called the User Agent. Further improvements and developments of the broader, 'abstract' architecture are then still possible, to investigate the feasibility of a society of autonomous, freely cooperating agents, which can still be seen as an applied research field in AI (c.f. [Schütz, 1990a]).

The complexity of the intended EUROTRA-II system implies the requirement for a layered, object-oriented approach for the overall design. We therefore propose an architecture for the EUROTRA-II development system which is based on the following five layers:

1. The User Interface and Presentation Layer
2. The Control Layer
3. The Application Layer
4. The Object Repository Layer
5. The Storage Layer

Between some of the layers there are communication channels which are responsible for passing messages between various agents, for triggering different types of requests, for exchanging configuration, control and user data. We distinguish between channels that deal with X-based messages and requests (between layers 2 and 3),

and that using a programmatic call interface (between layers 3 and 4, and layers 4 and 5).

2.3.1 The User Interface and Presentation Layer

The User Interface and Presentation Layer provides the man-machine interface (MMI) to different toolbox agents (e.g. grammar editors, compilers, debuggers) of the EUROTRA-II environment. The MMI is a graphical interface which uses windows and toggle menus with iconic symbols to provide an intuitive environment.

The interface and presentation layer provides the user with graphical access to four types of basic utilities which in turn provide additional functionalities according to their realization at the control and application layers: configuration, browsing, editing, and debugging utilities.

The basic processing unit of this layer is a server for the video display unit and the related input devices, like keyboard, mouse and other pointing devices; it is called the MMI-server. For each utility the MMI-server provides start-up toggle menus, so-called toolbox defaults which are requested from the User Agent and the agents of the application layer.

2.3.2 The Control Layer

The Control Layer is the main piloting and monitoring level of the EUROTRA-II environment. It controls each user-defined task and it is the first level of the NLP/MT development machinery.

The processing unit of this layer is the User Agent. (UA). It is responsible for the initialization of the environment at start-up time, the first level of communication with the user, including global help facilities and the handling of emergencies, the management of system resources, the scheduling of application processes and the overall housekeeping of the environment. The UA also provides system-wide facilities such as editors and browsers of the object library (c.f. section 2.3.4).

2.3.3 The Application Layer

The Application Layer constitutes the level of actual tools which are based on the (family of) EUROTRA-II formalism(s)[1].

The EUROTRA-II framework imposes a basic three stage architecture on the process of translation: analysis, transfer and generation. Analysis and generation are performed by the virtual machine agent (VMA), while the transfer agent (TRA) that is seen as a separate toolbox agent, extends the operations of the virtual machine (c.f. section 3.3) to perform the transfer process. This architecture does not prescribe a single stage of analysis or generation, so that the EUROTRA notion of a stratified model can be maintained if required.

The EUROTRA-II formalism (c.f. [Alshawi *et al.*, 1991]) attempts to incorporate the desirable features of several different, new formalism approaches in computational linguistics in a unified and flexible way. The formalism does not embody any particular (computational) linguistic theory, but serves as a formal language within which it is possible to encode many particular linguistic theories. To support this view the design of the formalism has also adopted a layered approach. The formalism has at its lowest level (or core) simple unification-based constructs, i.e. elements that are considered to be uncontroversial and which can be efficiently implemented. The basic core is extended and enriched by (currently two) further layers. The second layer is composed of notational devices which can be compiled into the core language, for example, disjunction and linear precedence constructions. The third layer is an extended constraint language that cannot be compiled into the core language. This last layer is yet not fully defined, so that the formalism remains open to accommodate future enhancements. Future inclusions to the constraint language of the third level, like those of other levels, are required to be declarative, monotonic and effectively reversible.

The linguistic toolboxes of the application layer, although somewhat based on the EUROTRA-II formalism, are in principle applicable, i.e. generalizable, to other NLP/MT applications, e.g. machine translation, natural language interfaces to information systems, writing aids, etc., where different kinds of linguistic theories, e.g. LFG, HPSG, SFG, etc., and linguistic processing techniques, e.g. for parsing, generation and transfer, are employed.

The application layer comprises four built-in toolboxes:

1. The text-handling toolbox
2. The linguistic processing toolbox
3. The grammar toolbox
4. The lexicon toolbox

Different (specialized) agents are the processing units of this layer: the Text-Handling Agent (THA) for the text-handling toolbox; Editor Agents (EDA), Browser Agents (BRA) and Compiler Agents (COA) for the grammar and the lexicon toolbox; the Virtual Machine Agent (VMA), the Transfer Agent (TRA) and Debugger Agents (DEA) for the linguistic processing toolbox.

The internal problem solving capability of these agents is to some extent formalism dependent in so far as most of it is specifically designed for EUROTRA applications, at least at the current stage of the project.

Other applications which might be independent of the EUROTRA formalism are also located at this layer. However, the central monitoring function in such a non-EUROTRA application, e.g. a natural language interface to command and control systems, is then handled by an Application Agent (APA) which takes over the UA's tasks for this application.

2.3.4 The Object Repository Layer

Access to the storage layer is handled through the Object Repository Layer. It functions as a uniform interface layer between the application layer of the EUROTRA-II system on the one hand and the underlying storage layer on the other hand. Linguistic objects or texts can be

[1]The EUROTRA-II formalism was the subject of a separate, parallel design study (ET-6/1) that was carried out by SRI International, Cambridge, England.

passed to or read from the storage layer through the object repository, which maintains information about the resource type, e.g. text file, linguistic object, processing unit, configuration data, etc., its status, e.g. free, in use, disabled, etc., and location within the file/data base system.

Internally, the object repository is connected to the data base (DB interface), to the object library (OL interface), and to the operating system (OS interface). The DB interface provides access to the lexical data base, the OL interface gives access to object-oriented descriptions of user and system resources, and the OS interface gives access to linguistic data stored in Unix files.

The introduction of the object repository layer ensures the openness and extensibility of the EUROTRA-II system with respect to different kinds of storage systems. The main processing unit at this layer of the architecture is the Object Manager (OM).

2.3.5 The Storage Layer

The Storage Layer serves as a foundation where text files, extra-linguistic data, e.g. user and system configuration information, as well as linguistic data are stored, The linguistic data consist of lexical information, e.g. monolingual and bilingual lexical data, and grammatical information, e.g. monolingual and bilingual grammar rules, as well as of intermediate results from linguistic processes,

This information will be stored as independently as possible of individual applications, i.e. it will be of maximum generality and should be suitable for use in different applications, e.g. machine translation, automatic abstracting, etc., and with different (computational) linguistic theories.

3 EUROTRA-II Components and Functions

3.1 The User Agent

The UA acts as a smart interface between the user of the system and different linguistic tools and applications. Its main tasks are the following;

- Communication with the user, the X components and the underlying operating system, as well as with the toolbox agents necessary for a user-defined application.
- Configuration support, e.g. parameter setting, user choices, global linguistic definitions, etc., and maintenance of the user's general and/or problem-specific environment, e.g. task definition and task scheduling.
- Control over resources, e.g. different kinds of objects and their information carriers, including application agents, exception handling, errors and emergency actions, in particular for those agents that do not possess a user interface.

3.2 Text-handling Toolbox

The text-handling[2] toolbox is an intelligent interface between a document in machine-readable format and the linguistic applications. It performs the following tasks:

- Convert the external text format, like nroff, L^AT_EX, proprietary word processors etc., into the internal standard format, called EDIF (EUROTRA Document Interchange Format). EDIF is based on SGML, a standard document markup language; basic document elements, like sections, paragraphs, headings, tables, footnotes, etc, are recognized and marked-up.
- Display, check and edit the output of the text analysis phase; to do this, commercial SGML parsers and editors are going to be used.
- Allow for different text-based applications to run on EDIF files. Among those applications are: text storage and retrieval tools, concordances and usage patterns, and statistical tools, like frequency counts, word lists, etc.
- Recognize and disambiguate sentence boundaries, wordforms, punctuation marks, etc.
- Run text-based applications at, the wordform level: special EDIF markers will be used for patterns like foreign word, proper name, abbreviation and acronym, code and model, date, number, etc.
- Word analysis: this will cater for orthographic analysis as well as word segmentation. In order to perform word segmentation, the lexical data base will be consulted for possible morphemes and morpheme sequences. As wordforms will have to be normalized before the lexicon look-up, a kind of two-level morphology augmented by featural information fetched from the lexical rules will be implemented.
- Finally, the input for the Virtual Machine Agent will be produced. This includes filtering out all the information which is not linguistically relevant (this information will be stored in a separate skeleton file), converting all linguistically relevant information into features, according to user declarations, and creating the proper data structures.

3.3 Linguistic Processing Toolbox

The linguistic processing toolbox consists of several submodules which support the user during the testing and debugging phases of a NLP/MT application:

- The virtual machine (VM) provides the capabilities of an inference engine (unification and constraint solution) operating on typed feature structures. The basic operations performed are: parsing, constraint evaluation and generation. A VM driver is responsible for the monitoring of the operations and the management of (sub-) results for further processing.

[2] The Texthandling sub-component was the subject of a third study within the EUROTRA-6 operation; this study (ET-6/3) was carried out by SEMA, Brussels, Belgium.

- The debugger will offer several possibilities for supervising the VM operations (setting of spy points, interactive debugging facilities, etc.).
- A browser will enable the user to inspect the (sub-) results in different ways (textual and graphical). It will be possible to print and edit (sub-) results.
- Interactive disambiguation will enable the user to monitor and support the operations of the VM. This capability is foreseen because the user has knowledge about his/her grammatical and lexical modules.
- A transfer agent enables the user to specify a transfer-based MT application. Transfer rules of the EUROTRA-II formalism are dealt with in the grammar and the lexicon toolbox. This module is then a specific realization of the VM which is able to perform graph transductions. The debugger, the browser and the interactive disambiguation facility of the VM are also be available for the transfer module.

3.4 Grammar Toolbox

The grammar toolbox of the system consists of several tools for grammar writers. The main functions are:

- Grammar coding: The user will be able to write and maintain grammars based upon the rule types defined in the EUROTRA-II formalism, and to compile them for the VM. Grammar maintenance work will be supported by editors and specialized tools and menus for the encoding of macros and defaults.
- Grammar search: Browsers will allow different search strategies based on features and other specifications, e.g. the rule name.
- Grammar import, export and merge: The user will be able to import (sub-) grammars, to export them or to merge grammars. Duplicate and/or conflicting rules will be reported.
- Administration and version control: Users will be allowed to define and run their private grammars for special tests or applications. It will be possible to merge these versions into an overall grammar.
- Consistency checks: The user will be able to run wellformedness checks on (sub-) grammars, especially when import, export and merge operations are performed.
- Grammar administration: The user will be able to define his/her features (legal types, attributes, values). This in order to make grammar checks feasible and to support editing, compiling and searching.

The grammars will be stored in Unix files (source form) as well as in structured libraries (compiled form) handled by the Object Manager, Global declarations, i.e. features and types, will be stored in the linguistic data base, to avoid redundancies and inconsistencies.

3.5 Lexicon Toolbox

The lexicon toolbox of the system consists of several tools for lexicographers and linguists. The main functions are:

- Lexicon coding: users will be able to key in, edit, etc. lexicon entries, either as a whole, or with different views on a given entry. Coding will be supported by a menu system where lexical features and their values are presented in a mouse-sensitive way, similar to the lexicographer's workbench available in the current EUROTRA-I prototype system.
- Lexical search: users will be able to browse through the lexicon, following search patterns consisting of combinations of features and values, rule names, rule types etc. It will be possible to print and further process, e.g. narrow down, the results of a search.
- Lexicon import, export, and merge: users will be able to import other EUROTRA lexicons, to export them, and to be prompted whenever there are duplicate or conflicting entries. This is necessary as there may be several sites involved in the development and interchange of large lexicons.
- Administration and version control: users will be able to define their own lexicons, private sub-sets, etc. for special purposes. These private copies will be merged into the group's overall lexicon.
- Consistency checking: users should be able to run utilities checking if a lexicon is (syntactically and semantically) wellformed, consistent, etc.
- Lexical data base administration: users will be able to customize the lexical data base, based upon their requirements and linguistic declarations (features, types, key fields, etc.). Data base administration will be performed by a separate application.

All the aforementioned functions operate on the interface provided by the object repository layer.

3.6 Object Manager

3.6.1 Architecture

The Object Manager (OM) is responsible for the storage and maintenance of the relevant system and data objects. Objects can be:

- part of the Unix file system,
- named portions (sets) of a data base system.

Objects stored as files are source text files, grammars, intermediate linguistic results, etc., but also system tables, executable modules, documentation, descriptions of tasks and work items, etc.; objects stored in the data base are mainly monolingual and bilingual lexicons.

As already outlined in section 2.3,4, the Object Manager fulfils two basic tasks: on the one hand it allows the UA to locate, access and manipulate user and system resources by means of unique identifiers and high-level functions, e.g. open/close, create/delete, execute, etc., on the other hand it provides a library of (file and data base) input-output primitives, thus increasing the independence of the application layer from the storage layer.

The OM controls requests to the object repository, and passes them either to the file system or to the data base frontend. The DB frontend is part of the application

software, and is able to perform more sophisticated operations than a standard DB product; in particular, it is able to perform unification and subsumption operations of objects with DB entries.

The DB frontend relies upon the services provided by a DB backend, for which several alternatives are possible, at different stages of the project:

- Low-cost low-functionality routines, like C/ISAM, may be suitable for a public domain version of the system.
- Standard (E)SQL data bases like ORACLE, INFORMIX, etc. can handle and store linear feature structures quite efficiently; they are not suitable for complex linguistic objects.
- Object-oriented data bases like GemStone, ONTOS, O_2 etc. could do this, but there are no industry standards for them yet.

For the 1991-1993 version of the system, preference should be given to relational DBMS's featuring standard (E)SQL.

3.6.2 Services

The main tasks of the OM are the following:

- It ensures consistency system-wide, as it is able to lock objects which are currently used, to support version control and cooperative work. This allows for a truly multi-user environment.
- It supports task description and configuration: one class of objects is a so-called 'work-item' consisting of a processing function, input and output files, lexicon portions, rule sets and parameter settings, A 'task', as seen by the User Agent, is then simply a sequence of work-items.

4 Interactions between the EUROTRA-II Components

4.1 General

The layered approach adopted for the EUROTRA-II architecture is closely related to the X-protocol paradigm. The user interface and presentation layer is therefore the location of the X server, which is responsible for the window structure building and all surface related information.

The UA is a X client, but with the additional task of being a EUROTRA application server. The different tools of the application layer are also X clients: they communicate on the one hand with the X server, and on the other hand with the UA. The access to different types of objects is channelled through the OM.

The selection and activation of the various toolbox agents is handled by the UA. Apart from the UA, little or no real-time communication is currently foreseen between individual processing agents,

In this scenario a user-defined task is a sequence of work-items; a work-item may consist of:

- a processing function, e.g. a command line;
- a grammar file;
- a set of lexical resources;
- a number of additional objects, e.g. text files, configurations data, etc.

Once instructed by the user, the UA is responsible for building an appropriate task, i.e. a problem solving plan, with the support of the OM; it is also up to the UA to schedule the task and monitor its execution. In particular when it runs in background mode, for example, text-to-text translation.

In the following section we will briefly describe some of the key interact ions between the toolbox agents of the application layer and the OM.

4.2 Interactions

4.2.1 Text-handling Toolbox - Object Manager

The text-handling toolbox uses the services provided by the OM and the underlying storage layer in two ways:

First, it has to access the data it needs; among them there are:

- Input text files and output segmentation files.
- Sets of morphographemic rules.
- Additional files and tables used for its internal operations, like abbreviations used for sentence boundary recognition.

These data (resources) are passed (granted) to the text-handling toolbox by the OM and locked against possible changes.

Second, it has to access the lexical data base for word segmentation. Segmentation is a critical operation wrt. performance, as substrings of words have to be identified and looked up in the data base.

Like in other NLP/MT systems, the segmentation module uses a kind of "character tree" structure where 'legal' lexical entries are stored as leaves, with pointers to the data base entries they refer to,

During segmentation, the character tree is browsed for legal candidate morphemes, and the data base is then accessed. This reduces disc accesses to a bare minimum.

4.2.2 Grammar Toolbox - Object Manager

The grammar toolbox agents, e.g. a grammar editor, use the OM for access to grammar objects which can be of different sorts, for instance, rules, macros, defaults, type and feature definitions. Grammar objects are mainly stored in structured or plain Unix files, therefore access is handled through the operating system interface.

In addition to the standard Unix file handling capabilities, the OM provides further locking and versioning facilities, especially when multi-user access to the files is allowed.

4.2.3 Lexicon Toolbox - Object Manager

The lexical toolbox agents, e.g. the lexicographer's workbench, use the OM as a tool for getting access to the storage layer, which acts as a physical repository for their objects. They store and retrieve entries and sublexicons, using a set of high-level calls. A separate toolbox agent provides lexical data base maintenance and administration functions.

4.2.4 Linguistic Toolbox - Object Manager

The virtual machine agent (VMA) needs various sorts of objects, for example:

- Compiled grammars and declarations of types and features, as well as defaults and macros; they are loaded from Unix files and the data base.
- * Lexical entries; for each morpheme recognized by the text-handling component, and passed on to the virtual machine, a lexical lookup has to be performed.

The main problem, however, is the interface structure between lexicons as stored in the data base, and the EUROTRA-II formalism. While the formalism manipulates complex objects, using variables, complex values for features, etc., lexical structures are basically linear, with considerably less complex feature-value structures. Moreover, the lexical data base should support several different formalisms, and therefore be somehow formalism independent.

Therefore, it is proposed to use 'compilers' which convert (flatten) the objects used within the VM into a linear structure based on a typed feature logic (TFL) language (c.f. [Schütz, 1990b]). It can be shown that these structures (terms) are mathematically equivalent to highly structured graph-based descriptions, as used in the EUROTRA-II formalism. The reverse operation (deepening) is also performed by these compilers.

This schema caters for a generic data base interface, and supports the EUROTRA-II formalism effectively ([Schütz, 1991]).

The transfer agent (TRA) has to consult the bilingual lexicon for certain rules or feature combinations, and to instantiate appropriate target language patterns. Lexical lookup is performed in the same way as described above.

5 Development Phases

For the development of the EUROTRA-II system, the Commission is considering the following implementation stages:

- a Mock-up Phase (1991),
- a Development Phase (1992-93),
- a EUROTRA Phase-in Phase (1993-94),
- a Shared-cost Phase (1994 and beyond).

5.1 Mock-up Phase

During this phase, the critical components of the EUROTRA-II linguistic processing toolbox, i.e. the VMA, will be prototyped by the EUROTRA software team, along with some of the User Agent capabilities. This phase also includes an assessment of the computational efficiency and the linguistic adequacy of the new formalism, as well as the low-level design of underlying algorithms and data structures for the linguistic processes.

At this stage, the object manager of the object repository layer might be implemented as a program which simulates data base capability and is completely loaded into main memory.

In parallel, the Commission will perform an evaluation of products and development tools to be adopted for the subsequent development phases.

A call for tender for the following development phases will be launched by the Commission during the second quarter of 1991.

5.2 Development Phase

This phase consists of two development stages and will produce an operational software environment, including a graphical user interface, providing the facilities required by the EUROTRA community,

The main features of the first development stage of the prototype are:

1. An overall user environment, including editors, browsers, etc.
2. A revised and augmented linguistic processing toolbox with essential debugging tools.
3. A basic text-handling subsystem.
4. An early implementation of the lexical data base component (depending on priorities and resources available).

The basic objectives of the second development stage (1983) are:

1. The integration of a full-fledged data base component,
2. A more sophisticated implementation of the text-handling component.

This stage adds multi-user capabilities to the system, therefore the object manager may require a re-implementation. The other components may have to be enhanced as well, with a view to making them fully suitable for development work within EUROTRA.

5.3 EUROTRA Phase-in Phase

At this point in time (late 1993), the system will be handed out to the EUROTRA researchers and will be made available to other project teams, researchers and system suppliers, in order to have a broader basis for assessment, experimentation and evaluation of possible further developments. With respect to this, the Commission is considering the possibility of producing a public domain version of the EUROTRA-II system.

At this stage of the project, the implementation work will concentrate on the distribution of the system across a network of Unix workstations and servers.

5.4 Shared-cost Phase

After the EUROTRA-II system has been adopted by the EUROTRA community and favourably evaluated by other project teams, we propose that the Commission should consider the possibility of undertaking a further development cycle, with a view to turning the system into a truly general purpose NLP/MT platform and workbench.

By this we mean a comprehensive 'professional' development system for designing and building research prototypes and pre-competitive applications. This could comprise:

- Toots for implementing different grammar formalisms.
- Tools for defining lexical structures and their schemata.
- Interfaces with machine-readable and machine-tractable dictionaries and other linguistic resources and environments, e.g. interfaces with text storage and retrieval systems, importation and exportation of lexical, textual and grammatical resources.
- Tools for building specialized applications.

The functionality and design of such a system would he largely derived from the experience made with the EUROTRA-II environment development. At this point in time, the software industry should he interested in such an enterprise, so that the development work could be undertaken on a shared-cost basis.

6 Conclusions and Perspectives

In this paper we have outlined the global architecture of the EUROTRA-II system and its main operational and functional components.

The design of this architecture is open and can therefore be generalized to encompass other natural language processing applications, but it also provides support to the EUROTRA machine translation philosophy while ensuring the reusability of lexical and grammatical resources implemented so far in the EUROTRA project.

Furthermore, this design allows further developments towards a highly distributed, cooperative system and the use of different additional resources, e.g. knowledge representation systems for specific domain dependent information, which would be handled by specialized agents of the application layer.

Acknowledgements

Thanks are due to the EUROTRA Commission team, especially Serge Perschke, project leader, Giovanni B. Varile, supervisor of the ET-6/1 formalism study, and the EUROTRA central software team. Thanks are also due to Klaus Netter, DFK1 Saarbrücken, Ralph Meyer and Hartmut Krasemann from CAP GEMINI SCS Be-Com GmbH, Hamburg, and members of the SNI METAL team for various contributions to the ET-6/2 study.

Finally we would like to thank Johann Haller, head of the EUROTRA-D research unit, and Hans Uszkoreit, University of the Saarland, for scientific advice and continous encouragement. Remaining weaknesses are the authors' sole responsibility.

References

(Alshawi *et al.,* 1991] Alshawi, H., D. J. Arnold, R. Backofen, D. M. Carter, J. Lindop, K. Netter, S. G. Pulman, J.Tsujii, H. Uszkoreit. ET6/1: (Draft) Final Report. S.R.I International, Cambridge, England, 1991.

[CEC, 1989] Commission of the European Communities. Call for Tender EUROTRA-6: Feasibility and Design Studies within the Framework of the EUROTRA Programme. Technical Annex, Part I and II, Luxembourg, 1989.

[Gajek *et al.,* 1990] Gajek, O., K. Luks, R. Meyer, K. Netter, J. Schütz, G. Thurmair, A. Wehrmeyer. Requirements Study for a EUROTRA-II software environment. Internal Progress Report, Luxembourg, 1990.

[Gajek *et al.,* 1991] Gajek, O., H. Krasemann, Th. Loomis, K. Luks, R. Meyer, K. Netter, W. Ramm, J. Schütz, G. Thurmair, A. Wehrmeyer. Feasibility Study for a EUROTRA-II System. Interim Report, Luxembourg, 1991.

[Schütz, 1990a] Schütz, J. A Multi-Agent Architecture for a NLP/MT Development Environment. Ms., IAI, Saarbrücken, 1990.

[Schütz, 1990b] Schütz, J. An Architecture for Reusable Lexical Resources in a Multi-theoretical Environment, Ms., IAI, Saarbrücken, 1990.

[Schütz, 1991] Schütz, J. Towards a Lexicon Interface Representation Formalism. Ms., IAI, Saarbrücken, 1991.