

Using multi-user hypertextual glossaries in multimedia-based learning

Luigi Colazzo

University of Trento

Department of Computer Science and Management Science.

Via Inama 1, 38100, Trento, Italy

Tel. +39 461 - 882144, Fax. +39 461 - 882124

E.M. colazzo@cs.unitn.it

Marco Costantino

University of Durham

Laboratory for Natural Language Engineering, Department of Computer Science

Science Laboratories, South Road

Durham, DH1 3LE, UK

Tel. +44 191 374 2549, Fax. +44 191 374 2560

E.M. marco.costantino@durham.ac.uk

Abstract

In this work we present a object class named G_lossary. The instances of the class G_lossary (Glossaries) are specialised hypertexts for a multi-user didactic hypertextual environment. An object belonging to the class G_lossary is an aggregation of objects. Among the classes comprising the class G_lossary, the class D_efinition assumes relevant importance. Each glossary can include one or more objects of the class D_efinition. An instance of D_efinition is a specialised object that allows the display of the glossary's definitions. A glossary replies to the requests coming from other hypertextual objects in the environment by sending back to the calling hypertext copies of objects belonging to the class D_efinition. The copies received by a calling hypertext become components of the hypertext. A calling hypertext can reside on any authorised remote machine connected to th G_lossary server machine. Since the instances of the class D_efinition are themselves hypertextual objects, they can also include starting points of links and, therefore, they keep all their properties also when included in a calling hypertext. The class G_lossary is an example of what we consider didactic hypertextual materials. The objects belonging to the class G_lossary have been implemented in Openscript involving the use of the DB3 standard libraries. The dynamic word-stemming algorithm has been coded in a Dynamic Link Library and is written in C language. The G_lossary server is implemented using a standard Unix NFS server.

Introduction

Some of the literary works that compose a literature play a subsidiary role for the reading activity. Some of these literary works are dictionaries, glossaries, atlases, concordance tables, chronologies, etc. This category of literary works has a special role in the literary universe. In a metaphoric interpretation they can be thought of as doors of communication between corpora of different knowledge and between heterogeneous reading experiences. In this view, the atlas that one could have used in reading Giulio Verne is the same (from a morphological point of view) as that one could have used in reading Darwin's work.

In this work we consider a specific class of literary works: the glossaries. Glossaries are literary works normally used as reference books (usually appended to a literary work that considers arguments typical of a subculture in which a definition of a specialistic term is given). Sometimes, glossaries assume the character of independent literary work and, in this case, are commercially known as dictionaries. Glossaries are extremely useful in the study of a new corpus of knowledge. The authors empirically observed, during their teaching activity of a University course regarding basic Information Technology, that to each increase of the knowledge of specialistic terms of the discipline, there corresponds an increase in the autonomous study abilities of the students. Despite their usefulness, a certain degree of intellectual rigour is needed for a productive use of glossaries. The reason is that, generally, dictionaries and glossaries are extensive literary works and, often, laziness prevails the need of knowledge of the person. Users without a minimum degree of rigour tend to construct a temporary hypothesis of meaning for the unknown terms, hoping that in the subsequent passages the meaning of the unknown terms will become comprehensibly clear from the context of the discourse. In limited situations, this intellectual shortcut can have positive effects. However, in the majority of cases will not have positive effects and will cause: (a) a loss of time, since the context almost never clarifies what is already supposed to be known to the person using the specialistic term; (b) a source of confusion, since the incomplete hypothesis on the semantics of the terms leads to wrong deductions; (c) an obstacle for the complete comprehension of the text. Deeply observing this phenomenon, it is possible to notice that it is not only laziness that causes the readers not to look up the glossary. If this was the case, in fact, the availability of an electronic dictionary should direct the users towards a greater use of the glossary. However, this does not happen or, at least, does not happen in what we empirically observed. We think that, thus, the reason why glossaries are not widely used is of a different nature. Our point of view is that what disturbs the reader is the loss of his context. In the case of a printed text, the path that the user follows for consulting the glossary can be schematised as follows: (a) a term is recognised as unknown in the reading process of text X; (b) the reader suspends the reading process; (c) the dictionary is opened; (d) the unknown term is looked up; (e) the definition of the term is read; (f) the reader restarts the reading process of text X at the point at which it had been suspended. What happens is that the reader has to move from the context of text X to the special dictionary context. Thus, he moves from a context which is, in most cases, organised as a narrative (the context of text X), to a context organised as a list (the dictionary context) and vice versa. Computer dictionaries often simulate the same interaction functionality, resulting in a similar loss of context. For example, one can consider the following interaction: (a) calling the glossary; (b) activation of the option for searching a term in the dictionary; (c) writing the desired term in the dialog box; (d) giving the confirmation of the operation; (e) the original window is substituted by a new window containing the text of the definition; (f) reading the definition; (g) closing the glossary; (h) going back to the original text X. In the prototype we present, these aspects of the problem lead us to use a new way of interaction between the glossary and the reader of a hypertext. In our approach an object belonging to the class `G_lossary` is a dispenser of copies of objects belonging to the class `D_efinitions`. Thus, the glossary behaves like a server of definitions. The client applications become owners of the copies and they therefore inherit all the links created by the author of the definition. These aspects are discussed in detail in section 2.

Glossaries are literary works organised following a specific morphology. It is useful to maintain this morphology in the conversion from the printed version towards the electronic version. This choice allows for the possibility of a quick learning process of the functionalities of the electronic glossary by the users. From this point of view, we can say that electronic glossaries must be "trivial", i.e.:

- the user must be allowed to read the definition in direct or inverse sequence.
- the definitions must be accessible using an alphabetic index.
- the definitions must be accessible through the input of a search string.

This requirement of the glossary to be "trivial" is satisfied by our prototype. In fact, the user can browse through an object belonging to the class `G_lossary` in the same way as he would do with the printed version. Moreover, the definitions can be accessed through an index or by inputting a search string. These aspects of the program will be discussed in detail in section 3. The requisite of a metaphorical isomorphism between electronic tool and printed tool does not, however, necessarily imply the renounce of the innovative aspects of an artificial processor, restricting the electronic glossaries to their basic performance. Our point of view is that an electronic tool must offer functionalities not available with the traditional printed media. The possibility to display the definition of the terms without exiting from the context of what the user is reading constitutes, for us, one of these functionalities.

The objective of this work is to show the feasibility of hypertexts which are simultaneously public and private. This is essential in an environment meant to encourage the learning processes through the use of hypertextual technologies. A hypertext, in fact, can be used in different ways within a didactic environment. The hypertext can be used to learn, to search, to present information and, finally, to build other hypertextual works [Cola95]. In all these cases a hypertextual work must provide services to suit the applicative task for which it has been developed but must also provide a means of communication between the different co-operating subjects involved in the process and an interchange ability of roles which constitutes the main didactic potential feature of hypertextual technologies. J.J. Legget and J.L. Schanase [Legg94] suggest that the problem of the condensation of the hypertext can be tackled from two different points of view. The first one implies the exchange of organised works such as hypertexts between the subjects involved. Clearly, this leads to the problem of the translation of the exchanged hypertexts from one hypertextual system to another in case the two subjects are using two different software bases. The second approach, defined as "hypermedia in the large", is instead based on the idea of a shared network distributed over more than one machines at different sites. One can speculate that the co-operation can be realised following an alternative way, one of which can be a model definable as "distribution & diffusion". In this approach, in *the large*, an hypermedia work also serves as a distributor of components that can be reused within hypermedia work created with other development systems. If the distributed components are well standardised, the hypertexts built using these components keep a meta coherence between them, due to the architecture of the system *in the large*, a feature that can be exploited in the process of translating of a hypermedia work from one system to another. We could describe the hypertexts built following this approach using the terminology of [Garz94] as follows: the application "in the large" is built as a collection of nodes that, apart from being accessible, can distribute copies of nodes, copies of composite components or slots to a client hypermedia. The prototype presented in this work explores this alternative through the building of a collection called G_lossary and is at present limited to the exchange between hypertextual works created within the same development environment. The work is organised as follows. In section one we shortly present the main characteristics of the objects G_lossary and D_efinition assuming a single-user G_lossary installation. In section two we consider different approaches to the construction of dictionaries and different strategies for the implementation of their interfaces. In section three we analyse the different ways of interactions of an object G_lossary and we briefly describe the difference between a multi-user installation and a single-user installation. Finally, in section four some aspects of the implementation of the prototype regarding the word-stemming algorithm are analysed.

1. The classes G_lossary and D_efinition

The glossary presented here can be considered as belonging to the class of objects named G_lossary (a collection) and partially composed of "nodes" which, in turn, contain "composite components" (objects) D_efinition. Each instance of the class D_efinition represents the definition of a term stored in the glossary. A glossary, therefore, can be ideally seen as composed of an unlimited number of D_efinitions. A G_lossary also contains nodes belonging to the class I_index and objects of the class M_enu. The objects belonging to the class I_index allow the user to access to different alphabetical lists of the terms stored in the glossary, while the objects belonging to the class M_enu allow the user to choose between the different options available within the application.

An unlimited number of glossaries can coexist and more than one hypertext can have access to their data. Each glossary will share the properties common to the class G_lossary and will differ from the other glossaries only in the semantic domain to which the definitions belong.

Users qualified for using the advanced features of the objects G_lossary can vary the objects D_efinition adding new terms or modifying the contents of the definitions. This is possible, as we will explain in section 4, because the objects belonging to the class G_lossary use a dynamic word-stemming algorithm allowing users to interact with a G_lossary in two different ways:

- users can access a specific G_lossary and create, modify or delete objects of the class D_efinition;
- users can delete, create or substitute objects D_efinition from their client hypertext.

In addition, authorised users are allowed to create new instances of objects G_lossary. Users not qualified for using the advanced features of the objects G_lossary can nevertheless interact with it in two ways. They can either consult G_lossary or ask for its services. In the first case, they can navigate between the glossary's nodes. In the second case they receive, as a result of a request, a legal copy of an object belonging to the class D_efinition which becomes part of the hypertextual work of the reader. The G_lossary comprises a dynamic word-stemming algorithm which allows to associate each occurrence of a term to a unique stem. In this way, all the forms of a verb (e.g. loves, loving, loved, etc.) will be associated to the stem "love". The algorithm successfully processes all regular parts-of-speech of Italian and English. It can also be modified by authorised users who can change the list of the basic lexical objects which are used by the word-stemming algorithm; this modification allows the user to adapt the glossary to be used with other languages than English and Italian. An instance of the class G_lossary can thus contain terms belonging to more than one language.

The class *D_efinitions* is composed by "composite components" which we will, from now on term as *item*. The *items* contain the text val and graphic elements which constitute the definition of the terms stored in the dictionary. An *item* can contain any kind of data: text, graphics and other objets. This means the user is free to create any kind of definition of terms. All the elementary objects that can be used within the development system used for the construction of the glossary can be employed within the definition of a term; this implies that the properties of the elementary objects are kept when an item is copied. A authorised user is thus free to define the characteristics of the items and is not forced in any way to make use of predefined schemes. This freelance in the definition of items will probably have to be limited in the development of the future prototypes, because it is directly derived from the homogeneity of the development environment. In fact, in a non-homogeneous environment, in which the requests can come from hypertexts developed with a different system, a translation of the item in a format compatible with the client hypertext will be required and this will almost certainly result in a limitation in the choice of the elementary objects that can be used for composing the items. There are three sub-classes in the class *D_efinitions*: text, graphics and text-graphics. The items belonging to these classes are predefined to facilitate the insertion of definitions mainly based on text, graphics or mixed classes by the user.

An important feature of the class *D_efinition* is the possibility to associate information regarding the copyright with each definition. This is allowed either for the text of the definition, for the graphical images included in the definition or for both. This information is kept associated with the item even in case the item is imported by other client hypertexts, and in case the information is saved in a file by a user's request. Since the copies of the items are standard objects, methods that operate on these copies can be implemented in a client hypertext. An example of a function that can be implemented is the creation of a special *G_lossary* obtained as composition of copies coming from more *G_lossaries*, or the function that allows the user to capture and save in an ASCII file any text included in an item, automatically retaining the copyright information associated to the same text.

As mentioned earlier, numerous client hypertexts can consult the data contained in the glossary. The glossary will respond to the requests of the authorised client hypertexts by sending the *item* corresponding to the definition of the term requested. Once imported in the client hypertext, the *items* maintain all the properties of the glossary, adding some new controls not previously available. These controls allow the user to move and close the items within the client application. An unlimited number of items can be imported into a generic client hypertext from an unlimited number of glossaries belonging to the class *G_lossary*. The *items* imported maintain the properties of the class *D_efinition* such as, for example, the information of the copyright associated to the definition. However, they become integral part of the client hypertext.

If the starting anchor point of a link is associated to an object *D_efinition*, it is kept during the copying process. However, in the first prototype developed, a dictionary of botany, this functionality has not been incorporated on purpose. The mechanism is basically similar to what happens when a WWW document is copied: the anchors defined in the document are copied as well.

2 Dictionaries and their interfaces

Traditionally, machine dictionaries are classified into different groups according to various criteria. A first distinction, by Raffaelli and Raspollini [Raff87], classifies dictionaries in three groups.

- The first group of dictionaries comprises simple transcriptions of dictionaries originally available on paper. The information is often stored without an advanced structure, maximising the access speed to the data and minimising the memory occupied. Examples of this first group are the dictionaries used in the spell-checkers.
- The second group of machine dictionaries comprises the dictionaries based on selected, classified and coded information. In this case, the key element will be the *stem* and not a generic term (as it happens in the first group). The *stem* represents the elementary entity addressable, with the corresponding grammatical category.
- The third and last group comprises the dictionaries that are called *lexical databases*, since they provide a representation, which can be elaborated, of the entire lexical information.

Another possible distinction is between *Machine readable dictionaries* (MRD's) which are dictionaries available in a format readable by the machine, and *Machine - tractable dictionaries* (MTD's), which are dictionaries treatable by a processor and often appropriate for uses in the field of "natural language processing". A MRD can be compared to a dictionary of the first and second group of the classification made by Raffaelli and Raspollini, while a MTD is similar to one belonging to the third group.

The use of dictionaries in interactive environments and the reading strategies followed by a reader while consulting a text have been studied by numerous researchers: [Guth87], [March89], [Blac91], [Wrig90], [Blac92]. In particular, Black A., Wright P., Black D. and Norman K. [Blac92] identify four different ways of user - glossary interaction within a hypertext. In the first case, once the user has recalled the glossary by 'clicking' on the unknown term, the glossary will appear hiding the area previously shown on the screen. In this case, the reader is forced to abandon completely the reading context (figure 1).

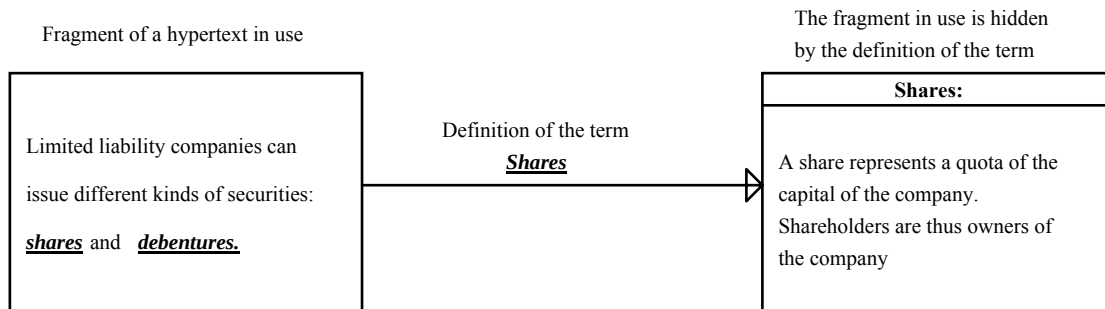


Figure 1: the glossary is placed on the text previously shown.

In the second case, once the glossary has been recalled, it will occupy only a part of the screen and, thus, the user will not have to abandon completely the previous reading context (figure 2).

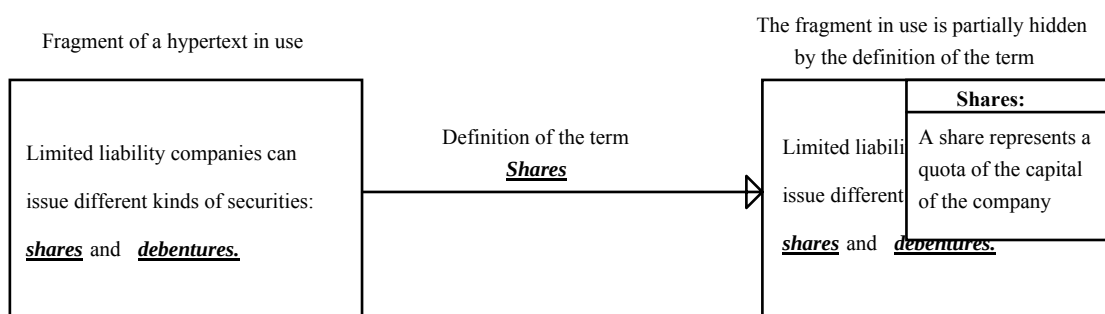


Figure 2: the glossary occupies only a portion of the whole screen.

In the third case, the glossary will be displayed in a small window, always visible, and placed in a predefined area alongside the portion of the hypertext currently displayed (figure 3).

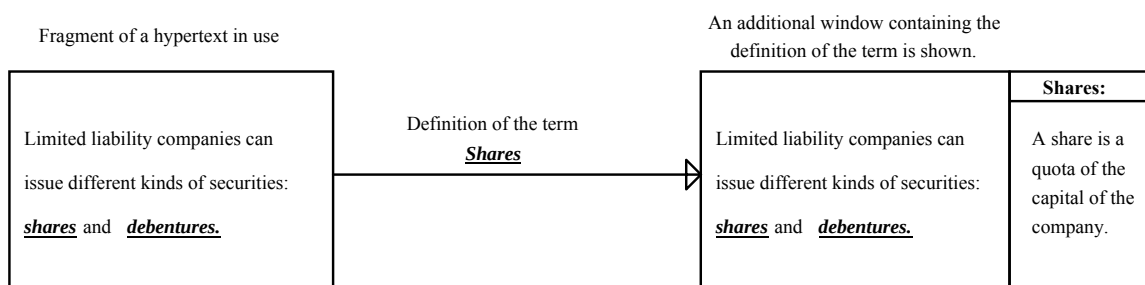


Figure 3: the glossary is displayed in a window separated from the portion of hypertext normally displayed.

In the fourth and last case, the terms stored in the glossary for the particular reading context will be visible in a window placed alongside the portion of the hypertext on display. The user will be allowed to recall the definition associated to the term by "clicking" on the unknown term shown in the list (figure 4).

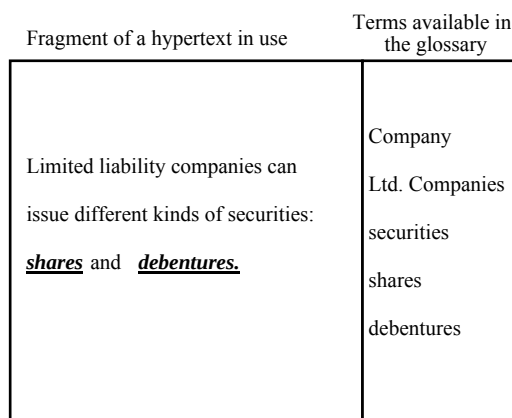


Figure 4: the terms stored in the glossary are shown in a list placed alongside the hypertext

Marchionini [March89], instead, studies how the behaviour of readers is modified in transition from a traditional printed encyclopaedia to the corresponding electronic version with hypertextual features. More precisely, Marchionini analyses how the *mental models* normally used by readers for consulting a traditional encyclopaedia are employed and how they are modified in the transition to the electronic version of the same encyclopaedia. Moreover, Marchionini analyses under which conditions the introduction of the electronic version of an encyclopaedia provides an improvement of the possibilities offered to the reader of the literary work. Marchionini's conclusions, reached using the printed and electronic versions of the *Grolier's Academic American Encyclopaedia* can be summarised in two points:

- the user of a traditional printed encyclopaedia normally has *mental models* sufficiently evolved for the successful consultation of an electronic version.
- moving from the traditional printed version to the electronic one provided good results but not optimal ones since the user, employing the *mental models* used for the traditional version of the encyclopaedia, ignored many search features available in the electronic version, using it in the same way as he would have done with the printed version.

In Marchionini's work, the development of electronic versions of encyclopaedias must help the adaptation of the user used to printed versions, providing similarities between the two. The user, thus, must be able to slowly adapt his mental models to the new features provided by the electronic version.

The glossary presented here does not follow any of the four approaches studied by Black A., Wright P., Black D. and Norman K. [Blac92]. The first of their configurations can in fact be obtained consulting G_lossary directly; in this case the node containing the object D_efinition can be placed over the window displaying the current text.

Approaches two, three and four (which is also available within G_lossary) share composition limitations, due to the fact that they require a fixed portion of the screen for performing the interaction with the glossary. Moreover, in all three cases, the objects that allow the display of the definitions also allow the display of the definition one by one. Finally, in solution number four, precognition of the requirements of the reader is needed. In the example shown in figure 6, the term *company* should not be shown in the list on the right, if we presume that the reader is sufficiently specialised, while it could be essential in a case where the reader does not have any knowledge of the specific terminology. Firstly, each term appearing in a text can be object of a search; the choice depends on the familiarity of the reader with the lexicon used. Moreover, the function carried out by a definition is prevalently temporary, since once the meaning of a term is understood, the definition is usually not needed anymore. The solution of having copies of the definitions allows the maximum freedom: copies of more than one definition can be requested by the user, the copies can be kept, can be iconised and, when they are not needed anymore, they can be destroyed. Finally, in our approach, each item has methods for moving it on the screen, which allows to overcome the composition limitations caused by having a fixed partition of the screen.

These features are available for the particular characteristics of the class D_efinitions. The glossary, however, follows Marchionini's suggestions [March89]. In fact, as already described in section one of this work, the glossary can be used in two different ways. A first way is to use the glossary as "stand-alone" application, while in the second case, the data stored in the glossary can be accessed using any authorised hypertext. In the first case, the user coming from traditional printed versions of encyclopaedia will be allowed to use the basic features of the program, using the "mental models" for the traditional version. Subsequently, he will be able to use the advanced search features and the options for recalling the definitions from a client hypertext.

3 Consulting and conversing with a G_lossary

As we briefly mentioned in the introduction, the objective of this work is to experiment didactic environments based on the hypertextual technology. In this perspective, we need an approach that allows for semantic consistency of the hypertexts developed, without transforming them into an advanced version of a programmed teaching system. An user that creates an instance of a G_lossary acquires the copyrights that he can eventually extend to others. An user that has copyright on a G_lossary can add, modify and delete the objects D_efinition of a specific glossary, either interacting directly with the G_lossary or interacting with any G_lossary's client hypertext. On the other hand, if an user has only reading right on the G_lossary, he will not be allowed to perform such operations, but only those regarding the consultation of the data stored in G_lossary and the requests of copies of objects D_efinitions. The consultation of the glossary can be performed in four different ways:

- stating the term to be retrieved in a dialog box shown using the specific option in one of the menus belonging to the class M_enun.
- selecting the term within any text of the glossary and pressing the right mouse button.
- selecting the term in one of the alphabetic lists available within G_lossary and belonging to the object I_index.
- following a Guided_Tour [Zell89], moving from a node to the next one or to the previous one in alphabetic order within a G_lossary.

The four kinds of search are notably different. In fact, while searching in the first and second way is performed using the word-stem algorithm, which can produce a negative response if the glossary does not contain the stem searched, the search using the alphabetical index is performed directly on the stems stored in the dictionary. In fact, if a definition is stored in the dictionary, then a node exists (a minimum destination unit of a link) which as name has that particular stem. An indexing algorithm automatically updates the alphabetical lists after each insertion of a new stem and keeps them ordered. The alphabetical lists of an object *I_index* are active. Selecting a stem and "clicking" the left mouse button in any of the alphabetical list causes a jump to the node that shares the same name with the selected stem. The same lists are used for handling the Guided Tour's alphabetic path. In case the request is made by stating a string or by selecting a term in a text within the glossary, the term is processed by the word-stem algorithm which extracts the stem which will be used as argument by the retrieval function.

This way of operation allows the user consulting a *G_lossary*'s *D_efinition* to use a portion of text that appears in the *D_efinition*'s text as argument of his search. This is particularly useful for concatenate searches which are potentially unlimited, since each of the definitions stored in the glossary can contain terms unknown to the user. We can consider, for example, a glossary of giuridical terms. The user, after having recalled the definition of the term *share*, will be allowed to recall the definition of the term *company*, shown in the definition of the term *share* and so on. These kind of searches are not rare among the users of glossaries.

Each generic hypertext can become client of a specific object belonging to the class *G_lossary*. To do so, it will only need to send a specific request to *G_lossary*, which will reply sending the methods necessary to the client hypertext to be enabled. These methods consist in the addition of a new menu to the client application for the interaction with a *G_lossary* and the primitives for the communication. The methods are sent in source code and eventually modifiable by the author or the reader of the client application. The transmission of the methods is done by sending a *composite component* having aspect of an icon. Once activated by pressing the left mouse button, the object installs the methods in the client application and self-destroys. The interaction features available within a client application reflect those available within the glossary and are:

- using *G_lossary* for consulting definitions of terms;
- stating the specific term to be searched in a dialog box recalled by a specific option in one of the menus;
- selecting a term within any text present in the hypertext and pressing the right mouse button.

The last type of search can be compared to the corresponding function in the glossary. The main difference is that, in this case, instead of executing a jump to the glossary node containing the definition of the selected term, an item coming from the glossary will appear, containing the definition of the term. This item will have functionalities normally not available within the glossary. In fact, when the copy is created, several methods are added for moving the item around the screen, destroying it, iconising it in the calling hypertext and, finally, moving the item to the destination node. The possibilities offered to the user-authors regarding the possibility to dynamically add new objects *D_efinition* dynamically is simplified by the particular dynamic word-stemming algorithm used. The algorithm differs substantially from the "static" algorithm normally used in glossaries and dictionaries, which, generally, do not allow the processing of the new terms inserted dynamically in the glossary. In this case, thus, the new terms would simply be treated in a different way from those stored at the beginning. Thus, using static word-stemming algorithms would cause considerable limitations in the possibility of updates of the glossary by a user. The algorithm presented here, on the other hand, is able to treat new terms in much the same way as the terms already stored in the glossary, without excluding them from the word-stemming algorithm and without the necessity of using a base lexicon. The new items will share all the properties of the class *D_efinition* and will be treated by the object *G_lossary* in the exact way as terms previously inserted and those stored at the beginning are treated. This leads to high flexibility.

3.1 Consulting and conversing with the *G_lossary* in a multi-user environment

The main different from a multi-user use of the *G_lossary* as opposed to a single-user user mode is that, in this case, the *knowledge-base* (the objects belonging to the class *D_efinitions*) are shared among all users. For example, if an user (e.g. a student in a classroom) updates the contents of the glossary with an improved or new definition, the knowledge will be immediately seen and shared with all the other authorised users (e.g. the whole classroom). Sharing the *knowledge-base* emphasises the *dynamic* approach taking in designing the class *G_lossary* presented in this work.

From a user's point of view, the introduction of a central *G_lossary* server does not make any difference. As described in the previous section, he will be allowed to browse in the definition of the terms and update the contents of the glossary either using the *stand-alone* mode or from a *client-application*. The only difference is that if the user is *authorised* by the system manager (*teacher*), any change he will make to the glossary will be immediately available to the all the other users.

In the previous sections, we described the system configured for a single-user workstation. However, the same considerations apply in describing a multi-user *G_lossary* system and the design of the system follows the same path. In the multi-user case a *master-copy* of each Glossary (belonging to the class *G_lossary*) is stored in the server's *file-*

system and is centrally consulted and updated according to the users' actions on the different machines. The only difference, therefore, is that the *G_lossary* is actually resident on a *remote* file-system, rather than on the user's machine.

To avoid conflicts, if a user modifies the contents of an object *G_lossary*, this is locked to avoid multiple modifications by other users at the same time. The other users enter a read-mode only, in which they are only allowed to browse the contents of the glossary without modifying it. Once the modification is finished, the new or improved *D_efinition* is immediately visible to all the users and the lock removed, allowing modifications by other users. The same applies to requests coming from *client-hypertexts* which, in an equivalent way, will communicate with the server's *master-copy* *G_lossary*.

The multi-user *G_lossary* system has been designed to allow users to build and update a common knowledge-base. However, it can also be used for more traditional uses, for example a case in which only the system-administrator (or lecturer) is allowed to make changes and update the *G_lossary*, while all the other users (students in the classroom) can use the *G_lossary* in read-only mode and are allowed to make changes only making a personal copy of the *G_lossary*.

4. Implementation

In this section we discuss in detail the implementation of the glossary and the algorithms used. The glossary here described is meant to be a generic object glossary belonging to the class *G_lossary*.

The glossary is written partly in *Openscript*TM and partly in C, as far as the word-stemming algorithm is concerned, which is compiled as a *Windows*TM *dynamic link library*. The program also uses the standard *Toolbook*TM library for handling database files in DB3 format. The client hypertexts authorised to access the data stored in the glossary include a module written in *Openscript*TM for the dialogue and the exchange of data with the glossary. The system is installable on any MS-DOS system with Microsoft *Windows*TM and Multimedia *Toolbook* 3.0TM. However, the suggested environment consists of a PC based on a 486 66Mhz CPU with 8Mbytes of memory and 1024*768 64k colours graphics card or better.

For a multi-user *G_lossary* installation a standard NFS server and NFS software for MS-DOS are required. The suggested installation consists of a NFS server running Linux or Free-BSD (at least a 486 with 4 Mbytes of memory and a 80M hard-disk) and public domain NFS software installed on each *Windows*TM machine.

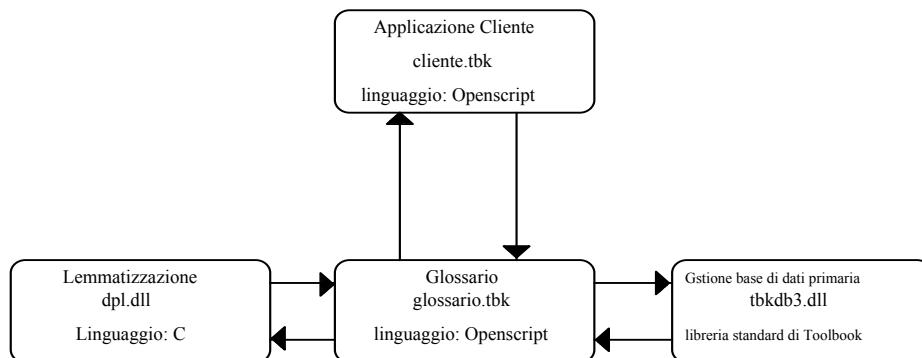


Figure 5: the main components of the glossary

The glossary responds to the requests coming from the internal user-interface or to requests coming from authorised client hypertexts (figure 5). More than one glossary object can be simultaneously active in memory. This means that the objects glossary must be able to identify the calling hypertext and address the appropriate replies to it, which consist of success messages, failure messages and, eventually, *items* containing the definitions of the terms requested.

4.1 The word-stemming algorithm

The *word-stemming* algorithm is written in 'C' language and is compiled into a *dynamic link library* external to the glossary. The algorithm used is *dynamic*. This mainly means that the algorithm is able to act autonomously, without any intervention of the user of the glossary and without the need of a support lexicon for obtaining the stem of the word. Moreover, the word-stemming algorithm is of a *morphological* kind, since the terms identified by the algorithm are not necessarily grammatically correct. The *word-stemming* algorithm performs correctly in the search process of terms already stored in the glossary as well as in the insertion of a new term and, in case the stem is already present in the glossary, the user is prompted. The drawback of the *dynamic* word-stemming algorithm is that it is not *exhaustive* nor *non-generative*. A *static* algorithm similar to the one used, for example, in [Raff89], would have provided better performance, but, on the other side, it would have meant the impossibility for the user to add new terms to the glossary

which would have been possible only excluding the new terms from the word-stemming phase and the impossibility to personalise the glossary for the use with languages different from Italian and English. Following our approach, the user must be allowed to create a new object G_lossary and use it to add an arbitrary choice of terms and, therefore, a static word-stemming algorithm would have been insufficient. The use of other kind of techniques such, for example, the technique of the *finite state automata* [Luc93] has not been considered adequate, because of the way in which some kind of functions, namely the one regarding handling some operations on the database such as the production of an alphabetical list of the terms stored in the glossary, would have been rather complicate.

The word-stemming algorithm is able to correctly identify the stems of all the regular parts-of-speech of Italian: nouns, simple prepositions, articles, prepositions linked with an article, adverbs, pronouns, adjectives, verbs, conjunctions. The algorithm uses three subsequent phases for searching a stem:

- a) The term indicated by the user is searched in the glossary. If it is not found, the length of the term is computed. In case it is less or equal to two, the term is supposed to be either an article or a simple preposition, which belongs to different stems and, thus, the searching process ends at the phase a); if instead the term's length is greater than two, the term is supposed to be another part-of-speech (noun etc.) and, therefore, the phases b) and eventually c) are entered.
- b) The last letter of the term is removed, obtaining a hypothetical stem, and the term is searched again in the glossary. This considers all the possible forms of the following part-of-speech: nouns, prepositions linked with an article, adjectives, pronouns.
- c) If the term is still not found, it can be supposed that, if it exists in the glossary, it must either be a verb, or an adjective in the superlative. Therefore, the proper word-stemming phase is entered: the term is matched against all the possible suffixes of regular verbs and superlative adjectives, ordered according to the length of the suffix. If a suffix is matched, it is removed from the term, obtaining the supposed morphological stem. The term is searched for the third (and last) time and, if the term is not found, the searching process fails.

The algorithm needs three searches in the glossary for verbs and superlative adjectives, whereas only two are needed in most of cases (for nouns). The word-stemming phase is freely modifiable by the user, who can change the suffixes used by the algorithm.

Each glossary belonging to the class G_lossary is based on two different databases:

- a fast access primary database which contains the list of the terms stored in the glossary, handled using indexed files in DB3 format;
- a secondary database which is kept within the glossary and contains, for each term, the data regarding the text associated to the definitions, the graphical images and other hypertextual objects.

The distinction between the two databases is due mainly to Toolbook'sTM slowness in handling accesses to data maintained internally to its collection. The primary database therefore acts as a filter to the secondary database which is accessed only if the requested operation (insertion, search, modification etc.) has been successfully performed in the primary database.

Conclusions

The prototype of hypertextual glossary we presented in this work can be thought as an example of what we consider didactic hypertextual materials: applications built as general objects which comply with the hypertextual technologies paradigms and which can be used as instruments during a learning process. The use of hypertextual technologies in learning processes, in fact, is not only concerned with the possibility of navigating in a collection of pages as it often happens in many hypertexts available on the market. If the usefulness of these technologies was limited to the retrieval of news only, the learning process would not be positively affected to a high degree. Some people could assert, not without any reason, that, since each real learning process is associated with a certain amount of work, what can be easily found is also quickly forgotten and, thus, all that reduces the amount of work involved counteracts the development of the learning. We think that these technologies show all their potential if used as instruments for the development of autonomous intellectual capabilities of the subjects of the learning process.

In the work we presented the prototype of a glossary usable as instrument in didactic environments based on hypermedia technologies. In section one we described the nature of the objects which compose the dictionary, in section two we compared the approaches followed by other realisations of the same kind, while in section three we described the way in which the users can interact with the glossary. Finally, in section four we briefly described the word-stemming algorithm used.

Bibliography:

- [Blac91] On-line consultation of definitions and examples: implications for the design of interactive dictionaries, in *Applied Cognitive Psychology*, pag. 149--166, No. 5, 1991.
- [Blac92] Black A., Wright P., Black D., Norman K., Consulting on-line Dictionary Information While Reading, in *Hypermedia*, pag. 145--169, Vol 4, No.3, 1992
- [Cola95] Colazzo L., Molinari A.: To see or not to see: tools for teaching, with hypertext slides, in *printing in EDMEDIA 95*, Graz giugno 1995
- [Garz94] Garzotto F., Mainenti L., Paolini P.: Adding Multimedia Collections to the Dexter Model, in *Proc.ECHT 94*, Edinburgh, 18-23 September, 1994- 70-80
- [Guth92] Guthrie L., Guthrie J., Wilks Y., Cowie J., Farwell D., Slator B., Bruce R., A research program on machine-tractable dictionaries and their application to text analysis, New Mexico State University, 1992.
- [Legg94] Legget J.J., Schnase J.J.: Viewing Dexter with open Eyes, in *Communications of the ACM*, Vol. 37, N 2, February 1994, 76-86
- [Luch93] Lucchesi C.K., Kowaltowski T., Applications of Finite Automata Representing Large Vocabularies, in *Software-Practice and Experience*, pag. 15/30, Vol. 23, January 1993.
- [March89] Marchionini G., Making the transition from print to electronic encyclopaedias: adaptation of mental models, in *International Journal of Man-Machine Studies*, pag. 591--618, No.30, 1989
- [Raff87] Raffaelli R., Raspollini C., Il sistema ABCD (A Basic Computer Dictionary): struttura e realizzazione di un prototipo di dizionario della lingua italiana, IBM Italia - Centro studi di Pisa, 1987.
- [Rumb91] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., Object-Oriented Modelling and Design, General Electric Research and Development Centre Schenectady, New York, 1991, Prentice-Hall International.
- [Wrig90] Wright P., Hull, Black A., Black D., Integrating diagrams and text, in *The Technical Writing Teacher*, pag. 244--254, N.17, 1990
- [Zell89] Zelleweger P., Scripted documents: A hypermedia path mechanism, in *proc. ACM Hyptertext 89*, Pittsburgh, PN, nov 1989.