

Introducing hypertextual glossaries in a multi-user didactic environment

Luigi Colazzo* & Marco Costantino**

*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

**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 Glossary. The instances of the class Glossary (Glossaries) are specialised hypertexts for a multi-user didactic hypertextual environment. An object belonging to the class Glossary is an aggregation of objects. Among the classes comprising the class Glossary, the class Definition assumes relevant importance. Each glossary can include one or more objects of the class Definition. An instance of Definition 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 Definition. The copies received by a calling hypertext become components of the hypertext. A calling hypertext can reside on any authorised remote machine connected to the Glossary server machine. Since the instances of the class Definition 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 Glossary is an example of what we consider didactic hypertextual materials. The objects belonging to the class Glossary 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 Glossary server is implemented using a standard Unix NFS server.*

Keywords: multimedia, hypertexts, glossaries, didactic environments.

Introduction

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 any positive effect. Observing deeply 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 therefore think that 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 since the user firstly recognises a term as unknown reading text X and suspends the reading process and, after having read the definition, goes back to the original text X. The sequence of actions taken in consulting a traditional printed dictionary and an electronic version are:

Printed dictionary: (a) the dictionary is opened; (b) the unknown term is looked up; (c) the definition of the term is read; (d) the dictionary is closed. **Electronic dictionary:** (a) the dictionary program is called; (b) the option for searching a term is activated; (c) the desired term is written in the dialog box; (d) the confirmation of the operation is given; (e) a window containing the text of the definition is opened; (f) the definition of the term is read; (g) the dictionary program is closed.

In the case of a printed text, 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. Thus the muscular effort is saved. In the prototype we present these aspects of the problem lead us towards the use of a new way of interaction between the glossary and the reader of a hypertext. In our approach an object belonging to the class *Glossary*¹ is a dispenser of copies of objects belonging to the class *Definitions*. 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. 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. 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 work is organised as follows. In section one we consider different strategies for the implementation of dictionary interfaces. In section two we short present the main characteristics of the objects *Glossary* and *Definition*. In section three we analyse the different ways of interactions of an object *Glossary* 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 Dictionaries interfaces

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.

The glossary here presented does not follow any of the four approaches studied in [Blac92]. In the first approach suggested by [Blac92], 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).

¹ The term *Glossary*, *Definition*, and *Index* in italic do not refer to the general notion of the term, but to the classes and objects used in our approach or to the glossary prototype.

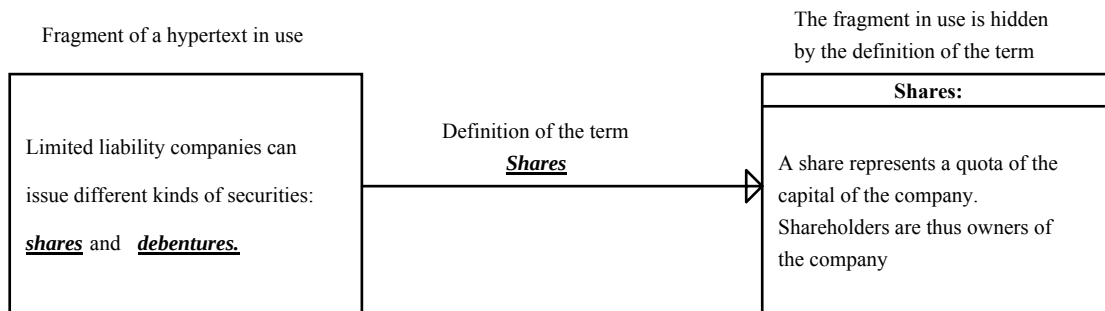


Figure 1: the glossary is placed on the text previously shown.

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).

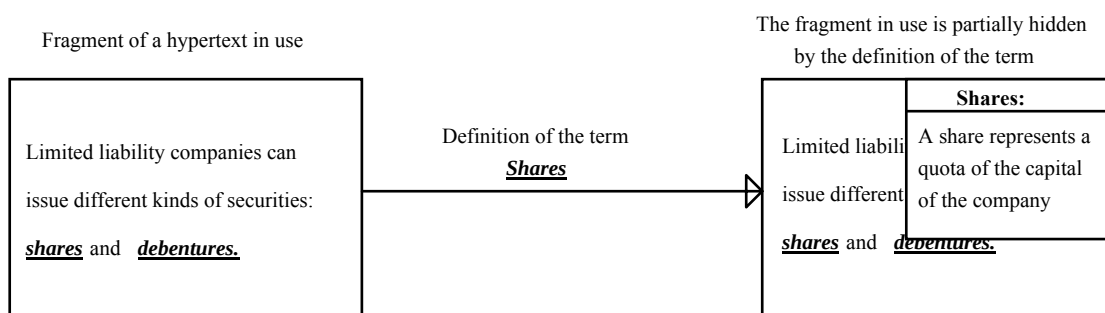


Figure 2: the glossary occupies only a portion of the whole screen.

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).

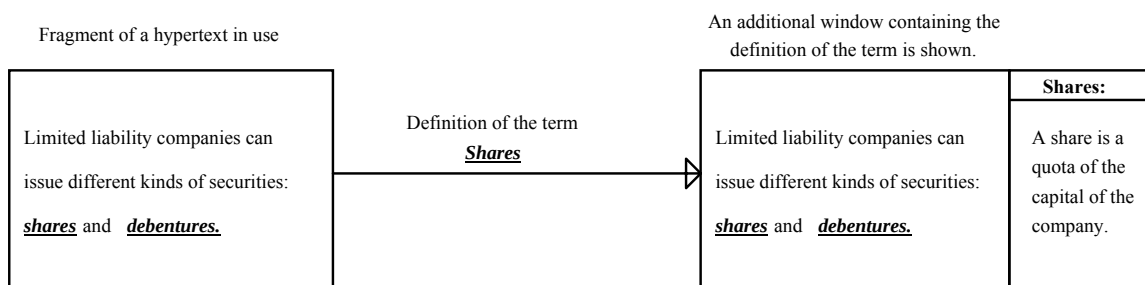


Figure 3: the glossary is displayed in a window separated from the portion of hypertext normally displayed.

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).

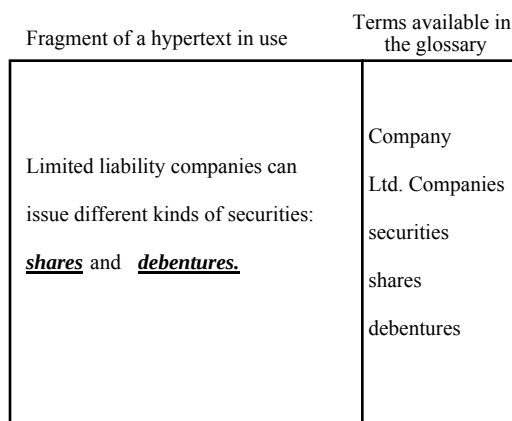


Figure 4: the terms stored in the glossary are shown in a list placed alongside the hypertext.

The approaches two, three and four imply composition limitations to the paging up of a hypermedia, because 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. In solution number four, precognition of the requirements of the reader is needed.

In our approach, instead, 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 *Definitions*.

2. The classes *Glossary* and *Definition*

The glossary here presented can be considered as belonging to the class of objects named *Glossary* (a collection) and partially composed of "nodes" which, in turn, contain "composite components" (objects) *Definition*. Each instance of the class *Definition* represents the definition of a term stored in the glossary. A glossary, therefore, can be ideally seen as composed of an unlimited number of *Definitions*. A *Glossary* contains also nodes belonging to the class *Index* and objects of the class *Menu*. The objects belonging to the class *Index* allow the user to access to different alphabetical lists of the terms stored in the glossary, while the objects belonging to the class *Menu* 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 to their data. Each glossary will share the properties common to the class *Glossary* 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 *Glossary* can vary the objects *Definition* 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 *Glossary* use a dynamic word-stemming algorithm allowing authorised users to interact with a *Glossary* in three different ways: (a) users can access a specific *Glossary* and create, modify or delete objects of the class *Definition*; (b) users can delete, create or substitute objects *Definition* from their client hypertext ;(c) users are allowed to create new instances of objects *Glossary*.

Users not qualified for using the advanced features of the objects *Glossary* can nevertheless interact with it in two ways. They can either consult *Glossary* 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 *Definition* which becomes part of the hypertextual work of the reader. The *Glossary* 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 *Glossary* can thus contain terms belonging to more than one language.

The class *Definitions* is composed by "composite components" here called *items*. The *items* contain the textual 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 objects. 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. An 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 freedom 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 *Definitions*: 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.

As mentioned earlier, numerous client hypertexts, even from different machines, 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 *Glossary*. The *items* imported maintain the properties of the class *Definition* 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 *Definition*, 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.

4 Consulting and conversing with a *Glossary*

The consultation of the glossary can be performed in four different ways: (a) stating the term to be retrieved in a dialog box, using an option available in the menu; (b) selecting any word within the text of the glossary and pressing the right mouse button; (c) selecting the term in one of the alphabetic lists available within *Glossary* and belonging to the object Index; (d) following a Guided_Tour [Zell89], moving from a node to the next one or to the previous one in alphabetic order within a *Glossary*.

The word-stemming algorithm (which can be customised by the user) is used in the first and second ways of searching and can produce a negative response if the glossary does not contain the stem searched. In the last two cases, instead, the search is performed directly on the stems stored in the dictionary. The Guided_tour searching method allows the user to use a portion of text that appears in the Definition's text as argument of his search and, potentially, for concatenate unlimited searches (for example, from the term *share*, to *company*, *liability* etc.).

Any generic hypertext can access to an object of the class *Glossary* by sending a specific request to which the *Glossary* will reply by sending the specific methods consisting in a new menu for the client application and the primitives for the communications. Once the setup of the client hypertext

has been completed, the interactions ways reflect those available within the glossary and are: (a) using the *Glossary* for consulting definition of terms; (b) submitting the term using a specific dialog box; (c) selecting any word within the text and pressing the right mouse button. Option (c) differs from the equivalent one included in the glossary for the fact that the definition will consist of an *item* imported from the glossary. The item will include methods for moving it around the screen, destroying it and iconising it.

The specific *dynamic* (substantially different from the usual *static* algorithms) word-stemming algorithm facilitates the dynamic insertion of new definitions by the user. Because of the particular algorithm, the new items will share all the all the properties of the class *Definition* and will be treated by the object *Glossary* in the exact way as terms previously inserted and those stored at the beginning are treated leading to an extremely high flexibility.

4.1 Consulting and conversing with the *Glossary* in a multi-user environment

The main different from a multi-user use of the *Glossary* as opposed to a single-user mode is that, in this case, the *knowledge-base* (the objects belonging to the class *Definitions*) are shared among all users, which emphasises the *dynamic* approach taken in the design of the class *Glossary*. From a user's point of view, the introduction of a central *Glossary* 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. Moreover, differently from the single-user version, *Glossary* is resident on a *remote* file system, rather than on the user's machine. Thus, a *master-copy* of each *Glossary* is stored in the server's *file-system* and is centrally consulted and updated according to the users' actions on the different machines. A file-locking system resolves the possible conflicts originated by the multi-user environment.

5. Implementation

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 3.0*TM. For a multi-user *Glossary* 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.

5.1 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 *Glossary* 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) technical terms, articles or simple prepositions; (b) nouns, prepositions linked to an article, adjectives, pronouns; (c) suffix-stripping for searching verbs and superlative adjectives. The algorithm needs three searches in the glossary for verbs and superlative adjectives, whereas only two are needed in most of cases (for nouns).

Each glossary is based on two different databases (stored on a central-server, for multi-user installations) for overcoming Toolbook's speed limitations: a fast access primary database which contains the list of the terms stored in the glossary (DB3 format); a secondary database kept within the glossary and containing the textual, graphic and hypertextual information of the definition.

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. 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.

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
- [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.
- [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.
- [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 Hypertext 89, Pittsburgh, PN, nov. 1989.