ELECTRONIC PUBLISHING, VOL . 5(3), 131–142 (SEPTEMBER 1992)

ALIVE: A distributed live-link documentation system

SILVANO POZZI
CEFRIEL
Via Emanueli 15
I-20126 Milano
Italy

AUGUSTO CELENTANO
Politecnico di Milano
Piazza Leonardo da Vinci 32
I-20133 Milano
Italy

LUISA SALEMME
Italtel
I-20019 Castelletto
di Settimo Milanese (MI)
Italy

SUMMARY

This paper presents the *ALIVE* project, which has been developed at Italtel to provide the means for automatic management of technical documentation. The main goal of *ALIVE* is to enable the user of a technical publishing system to establish live-links with data stored on remote databases. Live-links allow for automatic update of a document with database contents: whenever a modification occurs in the database data referenced from the document text, the document is updated accordingly.

The *ALIVE* user interface has been implemented by exploiting the functionality offered by the Interleaf[1] technical publishing environment, providing the user with a fully integrated environment. It allows the user to browse through a description of the available databases and to formulate queries related to data stored in them by means of a menu-based interface.

By enriching technical publishing features with data consistency controls in a uniform way, *ALIVE* moves towards the integrated desktop concept.

KEY WORDS Desktop publishing Hypertext links Relational databases Network communication User-friendly interface

1 INTRODUCTION

Technical documentation is the set of documents related to the management of technical information. With respect to office documents, technical documents exhibit a number of distinguishing features:

- they are structurally complex, being composed of several sections and subsections, with cross-references between them as well as links to other documents;
- they contain several types of information; text, drawings, graphs and tables highlight different document aspects or goals;
- they are usually made up from aggregations of data extracted from various repositories; in fact, the processes or products they describe are managed by different departments within organizations;
- they are updated according to changes in the process or product they describe; since the size of such documents is usually large, incremental updates, rather than complete reprocessing, must be provided.

[1] Interleaf is a trademark of Interleaf Corp.

CCC 0894–3982/92/030131–12

Received 28 January 1992
Revised 28 September 1992

Because of the composite nature of the underlying structure, a technical document can be modelled as a *hyperdocument*, that is, a collection of discrete pieces of data connected by links. As a result of the various types of the information described, technical documents also exhibit limited multimedia features.

In a large organization, the path to technical document preparation is traced by the associated industrial process. For example, R&D departments analyse the requirements for a new product, and design the prototype; the prototype is engineered by the Production departments, where changes are made for both technical and economic reasons; the Commercial departments package and distribute the product, and supply the user documentation.

In terms of the underlying hypertextual structure, the most evident consequence is that the hyperdocument nodes are originated by different departments within the organization, and reside on different data repositories. Therefore the producer of a technical document needs to locate the data in order to integrate them into the document. Moreover, the hyperdocument nodes are produced by different tools for two reasons. Firstly, the inherent multimedia nature of the information requires dedicated programs but secondly, in large institutions, new technologies are introduced according to need, rather than being the result of a thorough, integrated strategy.

Within the framework just outlined, two main problems influence the document production process:

- the consistency among the data extracted from different repositories at different times must be proved by the person who authors or edits the final document, since the data are not generated according to a unique consistent schema;
- the path to the physical location of the data on the network nodes must be known at each stage in which data need to be collected.

These two problems assume a great relevance in the management of technical documents that need to be updated whenever a modification occurs in a data source.

As an example, we examine the Hardware Faults Bulletin Board, which is used at Italtel to trace information about the unavailability periods of CPUs and storage units of installed computers, together with a description of the reasons for non-availability. Data regarding the various computers are entered in the databases of the Company departments on a monthly basis: the bulletin must be kept up to date with the last database entries.

Until now, the construction of a bulletin required producing a report from several departments' databases, and inserting it manually into the final document. This solution forces the bulletin's author to know where the data are stored, how to generate a report on the particular database system, how to transmit it along the communication network, how to import it into the documentation package used to produce the bulletin and, finally, how to translate it from the original format to the required document style. While each of these steps requires expertise usually owned by people devoted to the development of technical documentation, the whole process requires a mixture of knowledge and implies a set of verification steps which could be better managed by some kind of automatic processing.

An automatic system should satisfy the following requirements:

- it must be able to integrate data coming from different applications, by concealing the translation and formatting issues;

- it must provide automatic navigation capabilities, by means of an interface that presents the user with a uniform, logical view of the available data sources and which hides the details of their internal organization, query language and location. Moreover, it must enable the user to browse the information in several presentation formats: text, drawings, tables and so on;
- it must be able to create *live-links* from the document to the sources of the required data and must take care of the synchronization between the linked fragments. The live-link mechanism guarantees that, whenever a document is opened, the most recent version of the connected data sources is retrieved, and integrated into the document.

The remainder of this paper is organized as follows. Section 2 briefly reviews existing live-link approaches to document processing. Section 3 introduces the *ALIVE* system, exploiting the live-link approach in a heterogeneous and distributed environment, and illustrates its architecture. Section 4 describes the user interface, while in Section 5 the system performance is analysed and discussed. Section 6 contains the conclusions and some issues related to future investigations.

2 LIVE-LINK APPROACHES TO DOCUMENT PROCESSING

Many processing environments, and particularly those involving hypertext, documentation and integration, exhibit features which share the requirements stated in Section 1 and many are capable of exploiting live-links.

We assume the reader to be familiar with hypertext concepts [1–3] so we shall address only live-link related issues. In the hypertext paradigm, they come in the form of *warm* and *hot* links, that allow data to be exchanged across hyperdocument nodes [4].

To send data over the link the user must issue an explicit command. The contents of the anchor in the destination document will be replaced by the contents of the anchor in the current document. Similarly the user can issue a command to receive data through the link.

Hot linking augments this integration by providing automatic synchronization of linked anchors. In hot linking, an anchor is specified as the “master” and the other anchor is specified as the “instance”. Whenever the master anchor is updated, the new contents are sent over all the links attached to the master to replace the content of the connected instances. Conversely, when a document containing instance anchors is opened, the links are traversed to retrieve the contents of the master at the other end.

Tioga [5], the resident document editor of the Cedar environment, has been used as a testbed to implement *active documents*, that is documents with associated behaviours, called *activities*. In Tioga, documents are represented as trees of nodes, enforcing a logical structure. Activities associated with nodes are triggered when the contents of those nodes are requested. This allows the building of parts of a document based on other parts (e.g., a table of contents), or building a document as a collection of existing fragments, or including data from external database through a query mechanism.

In the Quill document editor [6] procedures can be embedded that are automatically triggered on opening or scrolling a document.

In Microsoft Windows and OS/2 environments a mechanism exists for applications to implement warm and hot links, based on the Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE) inter-application communication protocols [7, 8] DDE extends an application by enabling it to use information from another program. OLE has been designed in order to overcome a set of DDE limits; all OLE-aware applications are registered in a central database, so that every application knows the communication details of the others.

In version 7.0 of the Macintosh OS the *publish/subscribe* mechanism implements the hot link approach. According to this mechanism a document which has been declared as "publishable" can be "subscribed" to by another document. Any change in the subscribed document will be mirrored into the subscribing document.

The main drawback of existing *live-link* implementations is that they do not tackle the problem of connecting heterogeneous environments. For instance, the MS-Windows and the Macintosh approaches can be used only among applications which share the same operating system. On the other hand, the ultimate goal of the *live-link* mechanism is the building of the *integrated desktop* [4, 9], through the cooperation of several applications running on different computers with different operating systems. The network communication issue must be heavily taken into account, to define a linking mechanism which shields the user from the physical locations of data.

3 THE *ALIVE* SYSTEM

The *ALIVE* system was developed in Italtel on top of the Interleaf technical publishing editor. The tool is completely integrated in the Interleaf user interface, thus resulting in a functional extension of the editor itself [10]. The user can browse through a logical description of the available databases, and can issue a query on the selected database by means of a menu-based interface; the query results are automatically imported into the document, and can be automatically updated when the database content changes.

Figure 1 shows the functional architecture of *ALIVE*. The user workstation running the Interleaf document processing system contains the User Interface Manager (UIM) and the Formatter (FRM) modules, and communicates via a Network Service module (NTS) with the remote Database Server Management Systems (DBS). The database servers are accessed through a gateway node which, in addition to interconnecting dissimilar networks, also maps logical database names to physical locations via a Database Name Server Management System (DNS).

ALIVE supports document management through two phases, the *live-link* creation phase and the *live-link* activation phase.

The goal of the *live-link* creation phase is to establish an active reference from the document the user is editing, to data coming from a remote database. Since the reference is originated in a text processing environment, this phase is split in two tasks: first a formatted report is obtained from the database and, secondly, a reference to the report is established in order to include it into the document. In this way, the problem of accessing the database is solved within the context of the database management system, allowing a neat separation of functions among the components of the whole document management system.

The goal of the *live-link* activation phase is to execute the reference to the data, that is, to include in the document's text an updated copy of the data report. In principle, this simply involves access to the database management system to create the report anew; in practice, as we shall discuss later, only out-of-date reports need to be recomputed, and

ep066f01.eps

Figure 1. The ALIVE functional modules

local copies are used to improve the efficiency when no modifications have occurred since the last access.

3.1 Live-link creation

This process is activated directly by user commands. The *Live-link* Creation module (LLC) offers a user interface so the user, during document editing, can browse through the logical description of the available databases, locate the data repositories which contain the needed information, and formulate a query, on the selected database, using a menu which shields him or her from the specific query language syntax.

The query can be defined as either *static* or *dynamic*. In the former case the query result is imported into the document. In the latter case, a *live-link* is established from the document to the database: the query, which constitutes the link content, is performed each time the document is opened and if the relations which constitute the query target have been changed since the last document access. In any case, a reference to the query is inserted into the document and both the query and the corresponding result are stored on the user's workstation disk. The reference indicates whether the link is static or live.

Query results and error messages are processed by FRM and passed to UIM. Query results received from the Network Service module are formatted according to the Interleaf features.

UIM interfaces with NTS to run commands related to either database browsing or to the query formulation and execution processes. Requests to consult the list of the available databases belong to the former case. NTS interfaces with the DNS module, which manages a database wherein are stored the database names, their associated textual description, their network address and the date of their most recent amendment. This allows UIM to hide physical location of data from the user. By contrast, query operations belong to the latter case: NTS forwards the query to the appropriate DBS, where the query is actually performed.

A user query execution can cause an error (for instance, when a field referenced in the query has been deleted or renamed from the database), in which case DBS notifies the error, and UIM enables the user to re-formulate the query. When no error occurs, DBS sends the query result to UIM. Moreover, every DBS informs DNS about the date of the last update of its own data. This information is then exploited at *live-link* activation time.

3.2 Live-link activation

The *live-link* activation process is triggered whenever a document is opened. The functions involved are mainly carried out by the *Live-Link* Activation module (LLA).

LLA scans the Interleaf document, searching for link references. Whenever a static link reference is found, the corresponding query result is located on the user workstation disk and imported into the document by means of the Formatter module. Whenever a *live-link* reference is found, LLA addresses DNS to retrieve the date corresponding to the most recent change to the database involved in the query. When this date is earlier than the last opening date of the document, the query is forwarded to the appropriate DBS, where it is executed. Otherwise, the corresponding query result is located on the disk and imported into the document. Such a procedure avoids useless accesses to the database servers, thus reducing the network load.

4 THE USER INTERFACE

The User Interface Manager gives the user interaction facilities. With its help the user can:

- browse the list of available databases and explore their structure;
- issue a query to a selected database, exploiting an interface which hides the query language syntax of the specific DBMS;
- refine the query formulation, until the query result conforms to his/her needs;
- import the query result into the document through a static or live link;
- change a *live-link* into a static one or vice versa.

In the following, the flavour of a typical user work session is presented, which illustrates the *live-link* creation phase.[2]

UIM is fully integrated with the Interleaf user interface. Thus the user interacts with *ALIVE* by using a set of extensions to the typical interaction tools supplied by Interleaf, i.e., tags, pull-down menus and dialog windows. The set of Interleaf predefined tags is augmented with the query tag: whenever the user wants to define a link he or she selects from the tag pull-down menu the "*query*" entry. A paragraph tagged *query* is created and an ensuing dialogue defines the content of the link. A menu, whose entries are the names

[2] Note that, although in the reported examples a relational database is used, *ALIVE* is able to interface with every type of database.

ep066f02.eps

Figure 2. The Shutdown relation structure

of the available databases, is associated with the tag. For each menu entry two choices are available: *Help*, and *Data from database*.

The “*Help*” choice activates a procedure which queries the appropriate remote database server and displays a window containing the description and the fields list of the selected database. In Figure 2 the available databases are *Consumocpu*, *Terminali*, *Emissari* and *Shutdown*. The database *Shutdown*[3] has been selected: it contains only a relation, whose fields are *Data*, *Nodo*, *Durata* and *Causa*.

The “*Data from database*” choice enables the user to formulate a query, by means of a menu-based dialog (Figure 3); in the example, since the *Emissari* table belongs to a relational database, the output of the dialog is a SQL query. A sub-menu whose entries are the database fields is available, together with the *Select-all* and *Delete-interrogation* entries. The former retrieves the value of all the fields (i.e., it is equivalent to a “SELECT *” SQL clause) while the latter can abort the interrogation. For each field the *Condition-on-values* sub-menu entry helps the user in the definition of the logical conditions on the values he or she wants to retrieve.

At the end of the query specification process, the user decides whether the created link is *static* or *live*. The *query* tag is changed into a frozen-rep tag in the former case, and in a *report* tag in the latter case. The query results are then imported into the document.

[3] The Shutdown database stores information concerning the down-time of the network nodes, together with reasons for the non-availability.

ep066f03.eps

Figure 3. The Condition-on-values sub-menu

The *frozenrep* and *report* tag menus share the same entries (Figure 4). Through them the user can see the attached query, modify or execute it, change the link status, and perform cut-and-paste operations on the tag. This feature allows the user to move or duplicate the link anchor in the document, without specifying the query again.

5 SYSTEM PERFORMANCE

Apart from ease of use, which is guaranteed by the interface and the automatic management of remote database access, the most critical quality index of the system is the Response Time (RT), defined as the time elapsing from the moment when a live-link is processed, to the moment when the query result appears on the workstation screen. This unavoidable delay can, in principle, menace the interactivity of the service unless suitable limits are put on its maximum value.

Our experience showed that, while most of the "normal" word-processing functions should be completed in a few seconds, acceptable figures of the overall delay for a *live-link* execution can range up to about thirty seconds. These figures depend, among the other factors, on the number of requests present in a document and on the user's degree of experience in working with remote databases for complex retrieval.

There are several factors contributing to the overall delay: incoming command interpretation, network access and command transfer, query execution, delivery of the

ep066f04.eps

Figure 4. The report tag menu

report through the network, and, finally, report formatting and display. Each task of the retrieval chain involves a delay, which impacts the overall execution time.

Task execution time is also heavily influenced by the degree of resource sharing, and the time-sharing environment introduces its own additional delays. We performed a comprehensive simulation study, which allowed us to obtain a complete characterization of the system.

The model shown in Figure 5(a) shows the most relevant delay sources as being the waiting queues between the system modules. The queues occur whenever a process has to wait for connection to, or the execution of, a module.

To evaluate the system performance in a manner independent from the overhead introduced by the network, we built a test environment in which the document processing system, the gateway and the database, including the name server, resided on the same machine. Figure 5(b) shows the resulting model, where the queues related to the network and database access are omitted since they are not relevant in evaluating the delay. In this test environment the response time is therefore mainly due to query execution and to text formatting.

In Figure 6 a plot of the distribution of RT measured in the two environments is shown. The dotted line is related to the analysis performed in the network environment, while the solid line is related to the dedicated machine.

ep066f05.eps

Figure 5. (a) The network environment model. (b) The dedicated environment model

Network environment: the Response Time ranges from about 30 seconds to about 2 minutes, with a mean value of about 50 seconds, depending on the load of the communication network, of the gateway machine and of the database nodes;
Dedicated environment: here the Mean Response Time is about 30 seconds but this lies within a narrower range than before. In fact, the response time varies from 25 to about 40 seconds, depending on the complexity of the query.

The mean time figures are related to the alpha-test release of the system, and to a machine whose configuration was tuned for running the Interleaf package alone. Therefore they represent a “worst case” measure. Sample runs on faster machines equipped with larger memory, which are more representative of those that will be used for running the system, resulted in improvements of two to three times.

The broader range of response times measured in the network environment bounds the usability of the system to the network performance. We can in fact assume that, in any

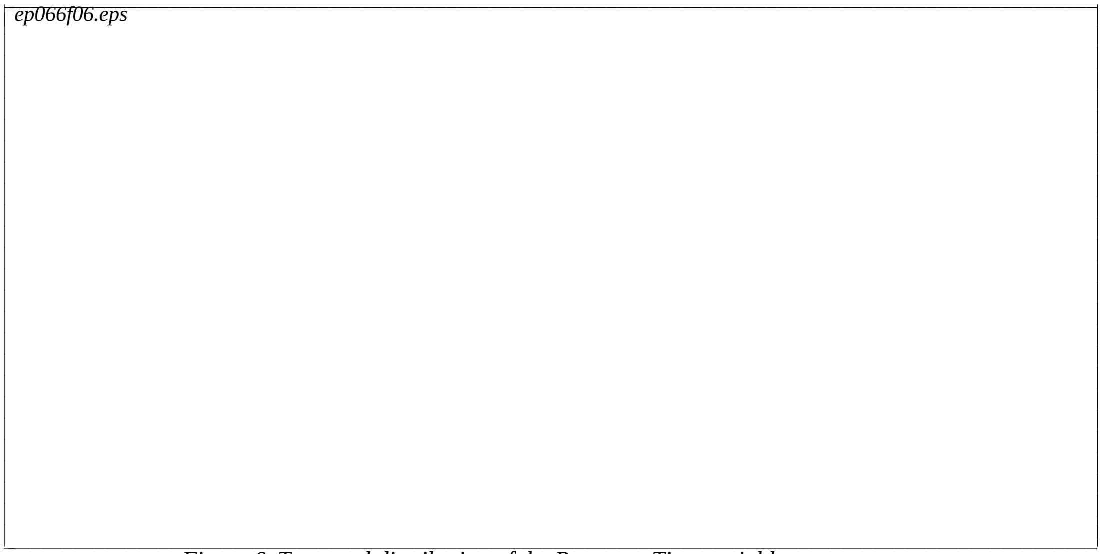

Figure 6. Temporal distribution of the Response Time variable

real situation, the performance of the DB server for collecting the data is not under the user control.

While in the best case the additional delay introduced by the distributed environment is not perceivable by the user, the longest waiting time could seriously constrain the effectiveness of the document preparation system. It is worth noting that this additional delay is caused not only by network access, but also by the login-logout procedures of the Oracle server. An improvement of the performance should also result from a more direct cooperation between the DB server and the document processing application.

Even without considering the improvement resulting from the use of a more powerful workstation, the time required to collect the data from the *ALIVE* environment is comparable with the time needed to access them with a direct database access. The automated data processing in the document body allows one to safely complete a document in a time which is noticeably shorter than with any partially automated procedure.

6 CONCLUSION

The *ALIVE* system enables the user of a technical publishing system to define documents whose contents are partly derived from external data sources. As the information provided by these sources changes over time, so should any document containing this information. Such features are supported by means of the *live-link* mechanism. In particular we considered the definition of live-links between a document and databases residing on remote nodes of a communication network.

The implementation of the *ALIVE* system was carried out in Italtel within the Research and Development division, whose environment is based on the Research and Development Communication Network. The network connects heterogeneous software and hardware platforms: VAX/VMS clusters, VAX/Ultrix and microVAX computers, Apollo, Sun, Hewlett-Packard and DEC workstations, and supports different kind of network protocols: DECnet, TCP/IP and SNA.

The user interface was implemented on a HP 9000 workstation by means of the Developer's Program Toolkit (DPT) supplied with the version 5.0 of Interleaf. DPT is a Common Lisp module which offers several procedures to manage a desktop publishing environment, by allowing customized menus, creating and managing menu entry lists, icons and windows.

The experimental activity done with *ALIVE* allowed us to identify some problems which will be the topic of future improvement, the most important being:

- the definition of a format which applies to the imported report, regardless of the data which is retrieved from the database;
- the extension of the *live-link* mechanism to manage connections to drawings and graphics.

ACKNOWLEDGEMENTS

We would like to thank Dr Flavio Mauri and Dr Rolando Madonna who contributed to the definition of the *ALIVE* architecture, and implemented the tool described here. We also thank Professors Maurizio Decina and Alfonso Fuggetta for their helpful comments and remarks. One referee suggested many improvements in the use of English.

REFERENCES

1. J. Conklin, 'Hypertext: An introduction and survey', *Computer*, **20** (9), (1987).
2. E. Berk and J. Devlin, *The Hypertext/Hypermedia Handbook*, McGraw-Hill, 1991.
3. S. J. DeRose, 'Expanding the notion of links', in *Hypertext '89*, Pittsburgh, PA, November 1989.
4. N. Meyrowitz, 'Hypermedia and the desktop of tomorrow', in *Proceedings of the International Convention on Hypertext*, Como, September 1989.
5. D. B. Terry and D. G. Baker, 'Active Tioga documents: an exploration of two paradigms', *Electronic Publishing*, **3** (2), 105-122 (1990).
6. D. D. Chamberlin, H. F. Hasselmeier, and D. P. Paris, 'Defining document styles for WYSIWYG Processing', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography (EP88)*, ed. J. C. Van Vliet, Cambridge University Press, 1988.
7. M. Vose, 'Hot links to go', *BYTE* (1990).
8. M. Heller, 'Future documents', *BYTE* (1991).
9. E. A. Bier and A. Goodisman, 'Documents as user interfaces', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography (EP90)*, ed. R. Furuta, Cambridge University Press, 1990.
10. R. A. Morris, 'The Interleaf user interface', in *Proc. of the Third International Conference on Text Processing Systems (PROTEXT III)*, ed. J. J. H. Miller, Boole Press, 1987.