UNIVERSIDADE DE SÃO PAULO

Authoring and searching in dynamically growing hypertext databases

PAULO C. MASIERO
M. CRISTINA F. DE OLIVEIRA
FERNÃO S. R. GERMANO
GLADYS PIERRI

Nº 06

NOTAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

Authoring and searching in dynamically growing hypertext databases

PAULO C. MASIERO
M. CRISTINA F. DE OLIVEIRA
FERNÃO S. R. GERMANO
GLADYS PIERRI

Nº 06

N O T A S D O I C M S C
Série Computação

São Carlos
jan. / 1994

Authoring and Searching in Dynamically Growing Hypertext Databases

Paulo C. Masiero
M. Cristina F. de Oliveira
Fernão S. R. Germano
Instituto de Ciências Matemáticas de São Carlos,
Universidade de São Paulo
São Carlos, Brazil
email: masiero@icmsc.usp.br
Gladys Pierri
Faculdade de Medicina de Ribeirão Preto
Universidade de São Paulo
Ribeirão Preto, Brazil

January 18, 1994

Abstract

In this paper we show how an application in the domain of Office Information Systems can be modelled so that a dynamically growing database of hypertext documents is created and automatically extended as well as easily searched. We propose a method for analyzing office applications which relies on a model based on statecharts to record the flow of documents within the system. A prototype implementation is described of a hypertext system to support the creation, storage and retrieval of documents associated to formal face to face meetings. Special features to be incorporated into hypertext systems aimed at supporting the storage and retrieval of office documents are also identified.

1 Introduction

The authoring of texts in hypertext systems and the searching of information in hypertext databases are activities which demand different skills and tools. In most applications the authoring process is performed only once, after which the hypertext document remains stable for long time periods, being searched (navigated, browsed) by end users with a range of possibly varied interests [1,2].

Existing hypertext systems provide support for creating text documents organised as nodes containing fragments of the text - sometimes called articles or cards - and a set of links connecting nodes that contain related pieces of text. Facilities are also provided for importing documents from external sources, a task which can be performed automatically if the text contains special symbols from a markup language. There are many ways to search through a hypertext database, for example, through a document index, or looking for particular keywords or yet browsing through a chain of linked document fragments [3]. Although in this paper we use the term hypertext to denote documents with text-only nodes, the reasoning presented here may be applied to similar applications dealing with nodes containing graphics or other media information.

This work is primarily concerned with the automatic creation and update of hypertext databases which grow regularly while being simultaneously used for searches in the intervals between updates. When a new text is inserted a potentially large number of new links must be established with the documents already in the database, a task which can hardly be performed manually. We propose a solution that takes advantage of the underlying hypertext system to support the insertion of new information into the database. This solution involves a static and a dynamic modelling of the documents within the system, and we believe it can be generalised to any application that requires the searching and insertion of information on expanding databases.

The structure of this paper is as follows. In Section 2 automatic hypertext authoring is briefly discussed, and related work within this context is reviewed. A method to analyse office information systems is proposed in Section 3 aimed at organising documents in a non-linear form which can be supported by hypertext systems and used to create and maintain a hypertext database. An office information system application containing a constantly growing database of interrelated text documents is described in Section 4.

That section also shows how this example was modelled, and describes a prototype system implemented according to the proposed model to support part of the authoring activity and the searching of information. It uses a commercially available hypertext system, integrated to a subsystem designed to act as a front end to the hypertext system and tailored to this specific application. This experiment has led us to identify some useful extensions which could be incorporated into hypertext systems in general. These are discussed in Section 5, and some concluding remarks are presented in Section 6.

2 Hypertext Systems and Automatic Authoring

Hypertext systems are receiving much attention as a new approach to organise on-line information, inspiring novel ways of accessing and using such information. However, the creation of hypertexts or the conversion of linear texts into hypertext format are not simple processes, and neither is the searching of information and navigation along the nodes of a document once it is created or converted. These are critical problems which, if not properly treated, can inhibit designers of considering the development of hypertext-based applications, and also discourage users of trying to benefit from non-linearly organised texts provided by hypertext applications.

Research on authoring is concentrated on two main lines: automatic construction of hypertexts, concerned with verifying which types of linear texts can automatically or semiautomatically be converted into hypertext and how this can be performed; and on the development of authoring tools to support the authoring of new hypertext documents. Two scales of authoring have been perceived: small-scale authoring and authoring-in-the-large, with the latter focusing on the development and long-term maintenance of complex hypertext documents [4].

Furuta and others in [5] consider that a hypertext designer must carefully identify those sources which are amenable to hypertext conversion. They identified several rules to assess the viability of a conversion. These are based on criteria such as the availability of a large amount of information organised in many fragments; how much the fragments are related among themselves; and whether the user will require only part of these fragments

at each access. They have studied how four different types of linear text can be converted into hypertext. The main issue on these studies was to discover the structure and the markers in the source documents. Documents with markers and with an easily recognizable structure, as well as with a repetition of structures, were more successfully converted than those not presenting these characteristics.

Glushko in [6], and Nunn in [7], have also identified factors relevant to a successful conversion, such as the existence of methods to characterise the physical and logical structure of the conventional documents and to identify exceptions embedded within these structures. They also emphasise the importance of analyzing how users retrieve the documents and which information is most important to them, in order to create meaningful links which will eventually prevent user disorientation. Following a similar line of work, a conversion of a text book into hypertext format was reported by Rada, in [8].

Other research projects draw on ideas from conceptual modelling and knowledge representation to create environments to support the authoring-in-the-large process. Their aim is the authoring of hypertexts characterised by databases containing complex documents, large amounts of information, collaborative authoring and maintenance over large periods of time [4]. The Concordia system is an example of an integrated environment for creating, publishing and maintaining complex documentation. It was developed to help the management of the Symbolics computer system's documentation [9]. One of its features related to hypertext technology is that links can be of different types allowing, for example, the insertion of text inside another piece of text and simple cross-reference.

ThyDoc is another environment developed to support hypertext document creation with authoring-in-the-large facilities. Its facilities include support for the creation of authoring links which are not explicitly shown to end users, defining underlying document structures e.g. composite nodes that, although visualised as a continuous flow of text and graphics, are internally formed by several nodes [4].

Any of the hypertext systems available in the market will offer tools for the authoring of hypertext documents. Usually these are small-scale authoring tools which include a text editor with an integrated spelling checker, a drawing package for the preparation of figures and text file importers. More advanced tools are ocasionally available, such as the automatic identification

of dangling references which must be completed by the authors [10], or facilities to rename nodes with automatic update of all associated links, thus minimising the problem caused by premature structuring [11]. Unfortunately most commercial systems available today offer only part of these features, and probably a system containing all of them cannot be designed within a single model of hypertext.

Our work also draws on authoring tools and conceptual models, following the same lines proposed by Stotts and Furuta, in [12], and Zheng and Pong, in [13]. The first authors use Petri-Nets as the underlying model of a hypertext system. The body of research and results already developed for Petri Nets, such as algorithms for node reachability, are thus readily available to be used at the hypertext interface level. The latter authors suggest the use of statecharts in a similar manner, exploiting their capabilities to model hypertext structure and browsing semantics.

We propose a conceptual model based on statecharts for describing the life cycle of documents to be inserted into a hypertext database. This model allows automating the process of preparing a set of text fragments to be imported by a hypertext system, thus providing for the automatic dynamic growth of the database. It also fully supports the users during the process of searching or browsing the database, offering meaningful paths which are very close to the mental models they create when performing such searches manually.

3 A Method for Eliciting the Structure of Office Documents

OADM, the Office Analysis and Diagnosis Methodology [14], models an office activity as a set of functions that manage and maintain resources related to the goals of the organisation. Each function handles documents which convey information about abstract entities (the resources) of an office and act in three distinct phases, namely initialization, management and termination. The initialization phase includes those procedures which create the documents that in the management phase will move through the many areas of the organisation, being assessed, authorised, printed, etc. until they are filed in the termination phase.

We consider an office information system composed of two main subsystems: a front-end subsystem that handles day-to-day transactions corresponding to OADM's initialization and management phases, and a back-end subsystem that handles the access to documents stored after the termination phase. The first one is a proper office information system while the latter is an associated system for information storage and retrieval which we believe can greatly benefit from a non-linear storage organization. One may view management functions and documents as belonging to orthogonal axis whose intersection determines the procedures (in the functions axis) and the atomic components they are applied to (in the documents axis).

The choice of a storage organisation to maximise posterior searches is determined by the path followed by the office documents during their active life and also by their relationships with other instances of similar documents. We suggest an organised approach for analyzing an office information system based on the documents handled by the system and their management functions, in order to determine a suitable hypertext database organization for storing the relevant information to be filed and eventually searched. Once the system is modelled, it is possible to create a hypertext database in which new links to documents already in the database are established automatically. The method, comprised of four steps, is described in Sections 3.1 to 3.4.

Our approach draws on data modelling techniques and, as usual in that area, we refer to types of documents. Later on, when referring to documents stored in a database, these will be referred to as instances of documents.

3.1 Step 1: Understanding the Application Domain

We start by analyzing the environment where the application occurs. The term application denotes the set of documents, functions and transforming procedures that characterise a problem area. In this step a taxonomy for the problem is organised as a list of relevant, domain specific terms within the system scope and their definitions.

A description of the problem may be used to produce an initial list of nouns which identify relevant actors in the system. One may follow, for example, an object oriented approach such as the one suggested by Booch, in [15], to identify the main terms related to the application. We are interested in documents carrying information about the resources managed by the sys-

tem; their originators and their consumers. Verbs are analysed to identify procedures that change the state of documents.

It is also necessary to establish a distinction between documents which are filed at the end of their life cycle and those that are simply discharged at some point. Documents which are filed at the end of their life cycle are further analysed in step 2. Our interest is to elicit their structure and to assess the viability of storing them as non-linear hypertext documents.

Temporary documents that do not persist between two consecutive system activations (or persist only for a short period of time), being discharged after originating some other (usually persistent) document, are seldom prone to benefit from a non-linear storage organisation, and are not further analysed in step 2. However, they are important for the underlying office information system that acts as a front-end to the storage and retrieval system, playing a key role in the analysis process of the latter. The analysis and design activities of the front-end system are out of the scope of this paper and we do not detail this aspect any further.

3.2 Step 2: Specifying the Atomic Components of Documents

Documents that are filed at the end of their life cycle are usually modified by procedures that change their status and possibly place additional information in the form of text or discrete data. It is possible to identify the atomic components of documents by analyzing the procedures that manipulate them. Each procedure changes some discrete data as well as unformatted text contained in the document; this chunk of data is a potential atomic component. Eventually, the identification of components may also be performed visually, as fragments of related data tend to appear grouped in visibly bounded areas on the document. For example, Figure 1 (Section 4.1) shows a document recording decisions of a meeting (minutes). Each topic discussed is clearly marked through session numbers, headlines, indentation and other visual clues; these topics are natural candidates to atomic components of the minutes.

As a result of this analysis, a static model of the atomic units comprising a document is produced, and for each of them a template is defined composed of four basic attributes: subject, resource, decision and type.

- The **Subject** is a short title (a single name) describing the contents of the text forming an atomic component. The name chosen to describe the subject should be related to the procedure that changes the document status and defines its life cycle (see next subsection).

- **Resource** indicates the resource or set of resources the document records information about. It is actually a meta attribute, composed of an attribute name and a possible list of attribute values.

- **Decision** is the final decision taken by the procedure applied to the document, which determines its life cycle, i.e., the path it will follow next.

- **Type**. A single document may be used to record information about distinct but similar procedures. The type attribute may be used to record this situation, thus avoiding proliferation of similar management functions. For example, a function that manages a request made by a graduate student to the office of a University President is the same, but for small exceptions, as the one that manages a similar request made by an undergraduate student. It is preferable to state the differences using an attribute attached to each atomic piece of document rather than specifying different functions for each possible variation.

The contents of an atomic component or of a composition of atomic components can be later implemented as articles in a hypertext-based application, and the attributes provide the key for establishing desirable paths for navigation. The decision on the links to be established is based on the generation of a model for the life cycles of the different types of documents, as described in Steps 3 and 4.

3.3 Step 3: Identifying the Life Cycle of Documents

A dynamic model for each type of document is then developed, relating its atomic components in a temporal order. The addition of atomic components to documents is determined mainly by the decisions taken at each stage of their life cycle by the functions in charge of managing them, which define the path to be followed by the documents at each point in their life. Thus,

each document may originate several life cycles, depending on the number of management functions related to it.

It is well accepted within office environments that most documents have a main flow path they normally follow. Eventually, exceptional situations may cause the early end of a function managing the document or, alternatively, may cause a detour in its main path, followed by a return to this path after some exception behaviour. A dynamic model will not necessarily contain labels corresponding to all the decisions allowed by the procedures dealing with an atomic component, as a function may prematurely reach its termination phase, after which it will not advance any further.

We suggest modelling the life cycle of documents using a modified version of statecharts [16]. The life cycle of a document may be represented by a set of statecharts where each atomic state corresponds to an atomic component of the document. The labeled transitions from each state indicate the possible paths that the document may follow from its current situation after a certain decision is taken. A transition label true indicates that any decision can lead to that path. We retain the history construct defined for statecharts with the same meaning of returning to the state configuration it was the last time the history target superstate was visited.

Decomposition of states is allowed, but with a different meaning than that in the original definition. XOR (exclusive OR) decomposition is used to create more succinct diagrams, saving transitions. A composite state represents a logical document (not necessarily corresponding to a physically existing one), comprising the subset of document's components represented by the atomic states within the composite state. Composite states lend themselves naturally to represent an interruption of the normal flow of processing. An example is a final state that can be reached from a number of processes after the occurrence of an error or cancellation event. An AND (orthogonal) decomposition indicates that the document's components within the orthogonal states are created or dealt with at the the same time by two parallel procedures. An illustration of such a diagram which refers to the application described in Section 4 is presented in Figure 4 (Section 4.2).

Each statechart specifying the life cycle of atomic components of a document provides a way of checking the consistency of every occurrence of an atomic component. This allows the implementation of algorithms for the automatic insertion of new documents in the database. As the components of a document are modelled as states in the statecharts, and decisions are

modelled as transitions, we must know which component of a particular document was the last one to be created and inserted in the database by the underlying office information system. Information about the instances of each active function is maintained in an auxiliary database containing pointers to the components already in the database to which a new component must be linked. It should be noticed that a new decision is known before the next atomic component related to the same function appears again to be inserted in the database. This allows the diagram to be non deterministic as we can always look backwards during insertion of a new atomic component in order to establish, from the set of possible decisions, which one was actually taken.

3.4 Step 4: Identifying the Links for the Hypertext Database

Based on the three previous steps it is possible to identify the links to be established when the first version of the hypertext database is created. These links will be automatically managed and updated each time the database is extended. We propose three link organisations: a temporally sequenced path (the natural path), and two sets of paths determined by the subject and resource attributes.

The natural path is composed by all the documents ordered by insertion date, and within each document, by a circular list of its atomic components. Another set of paths links components which are instances of the same function and components with the same subject across different instances of a function (subjects). The third set of paths links instances of occurrences of the same resource (resource type and/or resource values). The type attribute provides ways to establish the level of granularity desired for the two sets of non-linear paths, while the attribute decision is used mainly for integrity checking purposes.

An important decision to be taken at this step concerns the level of granularity of the links. This is a trade off decision as a coarser granularity will imply fewer paths to choose from and longer searches, whilst a finer granularity will imply in more paths and more complex procedures for automatically connecting the links. This decision must consider user needs and the capabilities of the hypertext system available.

As a final remark, we remind that other tasks follow the four steps described above which were not discussed. These include the assemblement, edition and maintainance of documents. We shall not dwell further on these tasks, apart from mentioning that the latter one will be automatic if the database is organised according to the approach proposed in this paper. We shall illustrate the use of the above method by describing how an example application can be modelled in order to be supported by a hypertext system and database.

4 An Example Application Involving a Dynamically Growing Hypertext Database

Storage and retrieval of information (texts, letters, minutes, forms, etc.) related to meetings is one of the most time-consuming office activities for clerks and knowledge workers. Many tools are used nowadays to support this activity, such as database management systems, report formatters and powerful graphical and text editors. They can be used in a stand-alone fashion or can be part of an integrated, comprehensive office information system. Hypertext systems can also provide useful tools in this environment. Recent research projects focus on collaborative work, i.e., providing support to groups of people working together towards a common goal. In [17], hypertext is suggested as a technology to help in the earlier stages of writing documents, and in [18] it is also being investigated as a tool to support face-to-face meetings for the joint preparation of presentations and proposals.

4.1 Formal Face-to-face Meetings: Description

The application chosen to illustrate the use of the approach described in Section 3 has been developed to support the difficult and time-consuming process of retrieving information from a file containing minutes of the meetings held by the Council of a School of Medicine which is part of a large Brazilian University. This University is composed of several units (Schools and Colleges), each of which has at least two Departments. Schools and Colleges have a Dean who presides over a local Council in charge of defining local policies and taking administrative decisions. Every School and College has representatives in a Senate at the University level, which is responsible

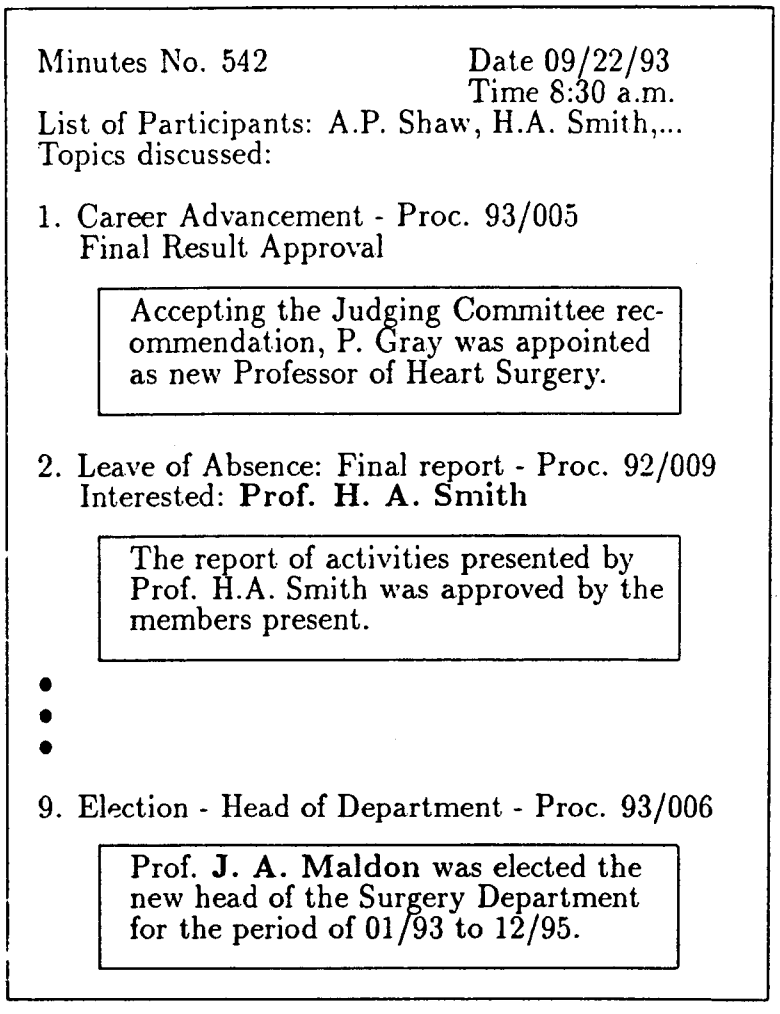

Figure 1: Outline of a Minutes

for the definition of global University policies, and is presided over by the University President.

Unit Council meetings are held once a month, when its members deliberate about important matters regarding long terms policies and also decide upon different types of request made by faculty members and students. These are called formal face-to-face meetings, being characterised by a previously issued and commonly agreed calendar of meetings, a summons issued by the chairman before each meeting containing the topics to be discussed; and a detailed record of the decisions taken during each meeting. A hypothetical example of a meeting minutes of the Council of the School of Medicine is presented in Figure 1. After being produced from tapes recording the meeting they are distributed to the persons who attended the meeting and a copy is filed (copies are sequentially bound in a book once a year). Currently a text processor is used to generate the minutes. A summons contains the date and time of the meeting it is calling for, and several topic headlines (that will eventually appear in the corresponding minutes).

Some of the Council decisions require discussions which span over several meetings and comprise several stages e.g. the opening of a position to hire a new lecturer requires an official announcement of the position, the assemblement of a selection committee, the choice of one of the candidates amongst those approved by the committee, etc. In many subjects of an administrative nature, it is usual that the requests made are similar to others which have already been analysed before, and the decision to be taken is better to agree with the one arrived at previously.

Consultation about the current stage of an ongoing process or about related decisions previously taken are the most common reasons for searching the minutes which record previous meetings. These searches can be quite complex due to the lack of a clear organisation of the recordings, apart from a sequential order by meeting dates. The clerk in charge of the searches has often only a few clues of what to look for. For instance, the name of a faculty member related to the subject; the type of request; or some vague idea about the date when a meeting where a similar decision has been taken was held.

4.2 The Approach Applied Step by Step

Term	Description	Observations
Session	Any meeting of the Council	Generates information to the minutes.
Topic	A particular problem or request the Council must decide upon in a session	Determines the atomic components of minutes
Summons	Contains a list of topics and is issued before each session, serving as a reminder and as a working document for members of the Council	It is not filed
Minutes	The record of a session, contains its corresponding summons and decisions taken about each topic during the session	It is filed
Process	An abstract concept represented physically by a folder or file containing related topics that appear in sessions spread over a certain period.	Corresponds to the abstract concept of a function in OADM. Its atomic components are managed by procedures
Meeting	Synonym for session	
Clerk	The office worker in charge of preparing the summons and the minutes	

Figure 2: Taxonomy of terms related to the management of Council Sessions in the Medicine College

Step 1 - Understanding the Application Domain

Analyzing the formal meetings held by the Council of the School of Medicine we learned that at each meeting (also called a session), its members deliberate about topics which are listed in the previously issued summons. The minutes form a document derived from the summons after the session, which reports on the discussions and decisions taken about each topic. Its contents must be approved by the members of the Council in the next meeting.

A fact that emerged after a more detailed analysis of this domain was that some topics appearing in several sessions and related by a common element (the management of the same resource, for example) are collectively called by the users a process, the same term being used to denote the folder where documents related to the process are kept. These actually correspond to the functions defined in OADM, i.e., they comprise the set of procedures that

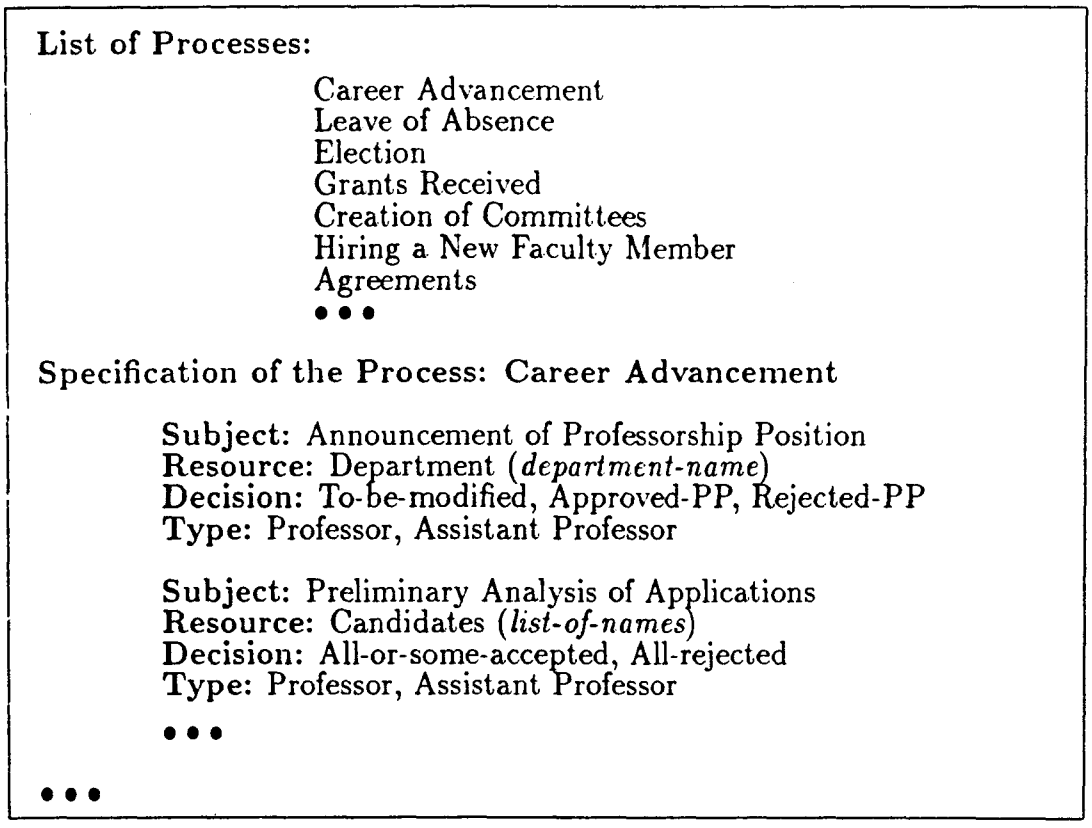

Figure 3: Specification of Processes

collectively manage the resources the Council is responsible for. From now on, we shall use the terms process and function as synonyms. In Figure 2 the taxonomy of terms for this example is listed.

It was decided that the system to support the creation, storage and retrieval of minutes of meetings held by the Council should handle two main types of documents, namely summons and minutes, of which only the latter are filed. Also, a topic in the minutes includes the headline which appears in the summons and additional text reporting the discussions and decisions taken during the Council session about that topic.

Step 2 - Specifying the Atomic Components of The Minutes

The different processes to be discussed in a session provide the key to determine the components of a session minutes. It has been seen that the atomic components of the minutes are determined by the topics in the summons, and

that the resources managed by the minutes are e.g. faculty members, departments and committees. Each topic admits a set of plausible decisions, one of which will be attached to the atomic component of the minutes corresponding to that topic. Some of the processes handled by the Council are shown in Figure 3, which also shows the specification of two atomic components for the Career Advancement process.

Any instance of the Career Advancement Process having a decision Rejected-PP when the Announcement of Professorship Position is discussed by the Council is forever stopped in that state; the same happens for a decision Accepted after a Request for Cancellation. An alternative design would be to explicitly have an abnormal or premature final state, and some events leading to it from some (super)states. The first solution was chosen because it closely models the reality of this application.

Step 3 - Identifying the Life Cycle of the Minutes

A total of twenty-six processes related to the minutes were identified in step 2 and a statechart was built for each one of them. Most of the resulting models – approximately 60 percent – were very simple, resulting in a flat, conventional state machine not requiring any of the advanced features of statecharts. In Figure 4 it is shown a statechart for the Career Advancement process. The path with no exceptions is composed of the states (atomic components) Announcement of Professorship Position, Preliminary Analysis of Applications, Composition of Judging Committee and Final Result Approval. A Request for Cancellation, for example, if accepted by the Council, prematurely interrupts the process; a rejection returns the process to its previous state. Consider also the transition labeled by the decisions Composed and All-or-some-accepted: after the decision of composing a judging committee has been taken and at least one candidate application was accepted, either Modification of Committee Composition or Final Result Approval will be accepted as new states for any instance of this process.

Step 4 - Identifying the Links for the Minutes Hypertext Database

Following the decision that a topic defines the atomic component for documents in this application we decided to represent each session minutes by an initial node containing general information about the minutes and a cir-

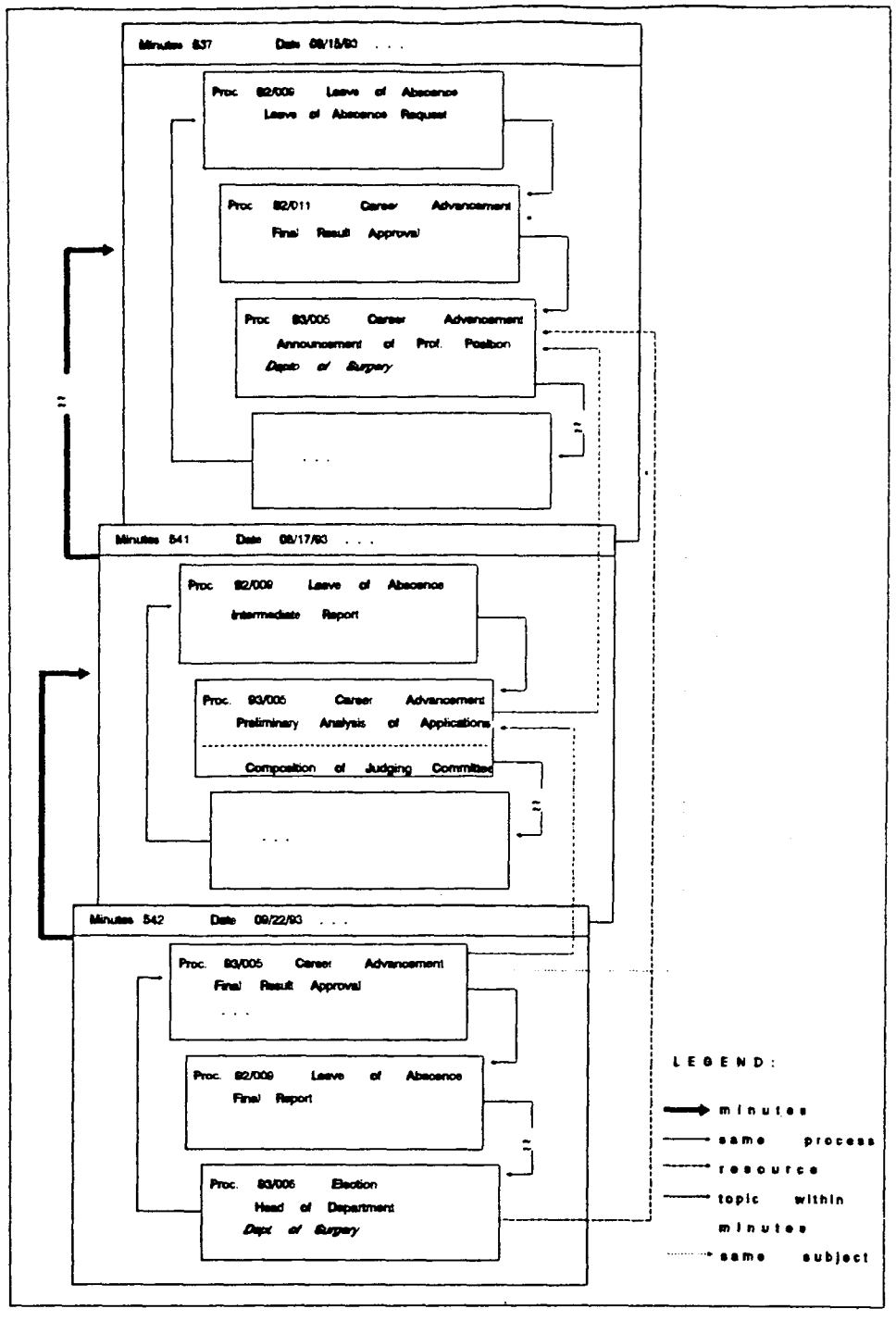

Figure 4: Statechart for the Career Advancement Process

cular list of topics. Other links are established according to the elicitation method: we want to link, for example, the topics related to a single process; topics with the same subject and topics related to a certain resource. We also decided for an intermediate level of granularity, differentiating according to process types and not linking instances of attribute values.

In Figure 5 we depict a schematic representation of these links for some hypothetical minutes. Note that not all the possible links are shown in the Figure. It shows, for example, that the atomic components "Proc 93/006 Election" and "Proc 93/005 Career Advancement" are linked due to the common resource Dept of Surgery, and also that all the atomic components related to the process "93/005 Career Advancement" are linked. Comparing Figure 5 with the statechart depicted in Figure 4 one may also verify that the latter process was concluded with no exception behaviour. A user can follow a path from the beginning, i.e., the most recently inserted article, and proceed to older ones following the track established by the links, as well as change to other paths whenever convenient. Additionally, a search by keywords or by an index of topics, if allowed by the hypertext system used, can provide the starting point for the user to proceed forward following the desired path.

4.3 Development of a Prototype of the Minutes Hypertext Database

A prototype of the system specified in Section 4.1 has been developed [19]. The system architecture shown in Figure 6 is composed of four modules. Two of them were developed specifically for this application and comprise a preparation subsystem, and two others support hypertext authoring and browsing activities, being implemented by Hyperties [10]. The version of Hyperties used could not resolve automatically any links of an imported document with articles already in the database, only those within the document being imported. Thus a program to help the user to semiautomatically convert these cases was also developed.

The creation of summons and minutes is supported by the preparation subsystem following the knowledge about the processes represented by the statecharts stored in a database. This database contains the generic life cycles of all processes, as identified in step 4 of the analysis method. An

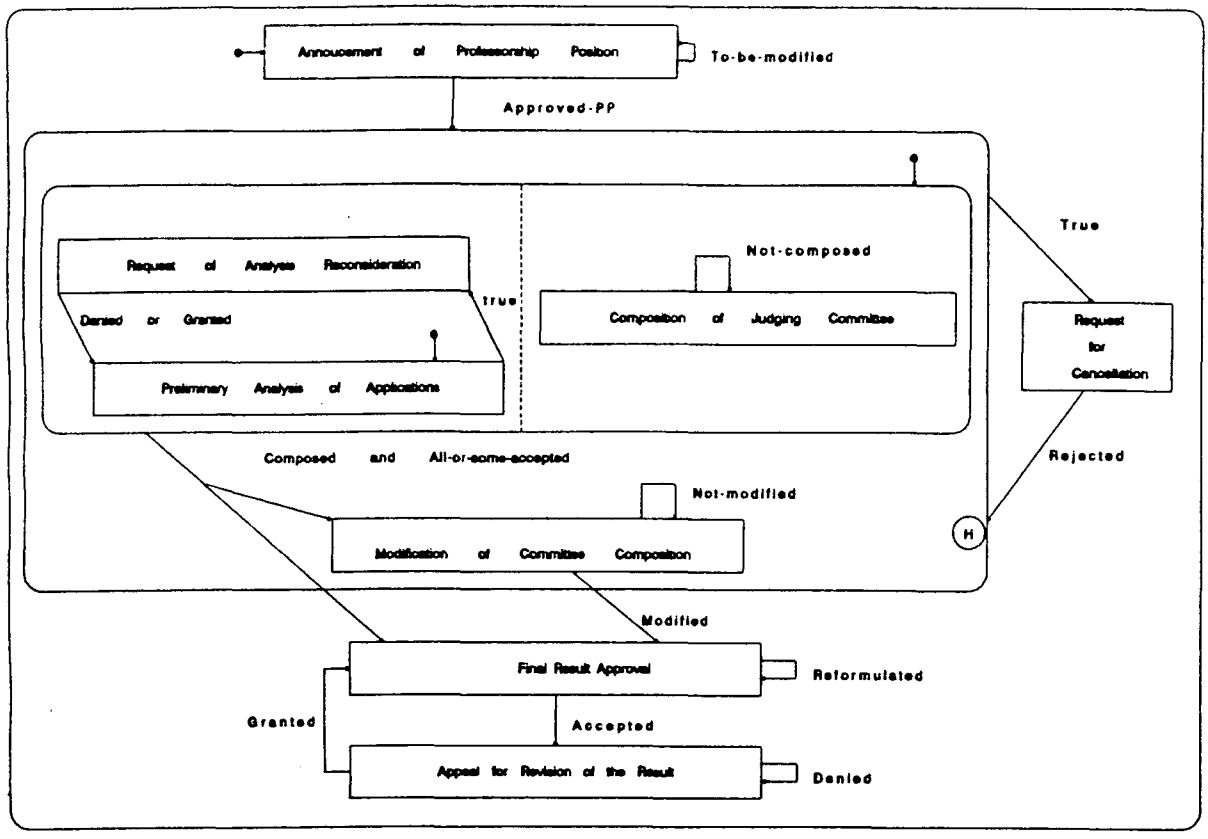

Figure 5: Schematic Diagram of the Hypertext Links

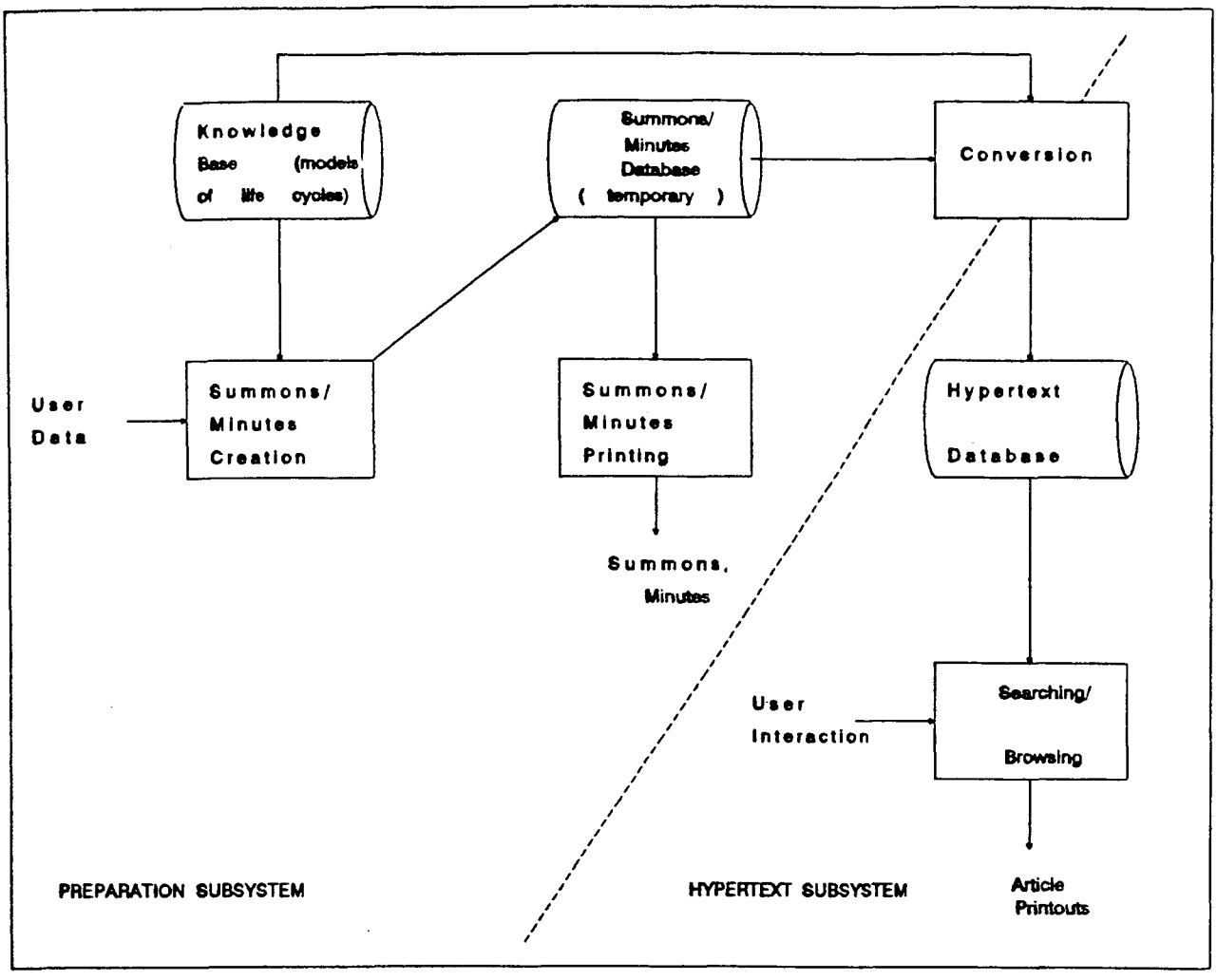

Figure 6: General Architecture

Minutes	*article*	—	—
Resource	Candidates Member of Committee Department • • •	*article* *article* *article*	— — —
Subject	Career Advancement	Professor Assist. Prof.	Announcement of Prof. Position Preliminary Analysis of Candidates Judging Committee Composition Final Result Approval • • •
	Election	Head of Dept. Dean • • •	*article* *article*
	Grants Received Creation of Committees • • •	*article* *article*	— —

Figure 7: Top Level Organization of the Articles

auxiliary table maintains the identifiers of each active instance of process in the database. In this way, when a new component is created all the links established in step 4 can be automatically specified in the Hyperties markup language and imported into the hypertext database. This subsystem also performs integrity checking, generates paper versions of the summons and minutes, and prepares the file with marked text to be automatically imported by Hyperties. It also generates a file containing any unresolved cases, to be handled by the user.

Searching and browsing are supported by the reading module of Hyperties, where all the links established may be followed. The interface is organised as follows: an initial article describes the contents of the base and its organisation, as well as instructions on how to conduct the searches. From this article the user can choose to follow the minutes (natural) path or to perform a search by subject or resource. Choosing resource leads to a choice of attribute type, whilst subject leads to a choice of processes. The search goes on until the actual topics start appearing as articles. Any links that can be followed from an article appear as keywords highlighted in the text or as special instructions at the bottom of the screen. The hierarchy of articles is illustrated in the table shown in Figure 7.

5 A Preliminary Evaluation of the Proposed Method

The prototype described in Section 4.3 was tested by end users with a real hypertext database containing several minutes, the testing being carefully monitored by the system developers. The procedure of importing new minutes was thoroughly tested, and many searching and navigation sessions were also conducted. These testing activities showed that the users approved the new way of organising and searching the minutes and learned quickly how to use the new system. However, several deficiencies of the prototype emerged, the analysis of which allows an improved specification of the system described in Section 4. We also found that some ideas adopted in this specific application, as well as some solutions proposed to address the problems encountered are applicable to hypertext systems in general.

With regards to the authoring subsystem two main points are worth commenting. First, the preparation subsystem is viewed by end users as a system completely independent from Hyperties, mainly because Hyperties (at least in the version used) cannot automatically link the markers in an imported file to articles already in the database. Secondly, users considered that the module responsible for handling the statechart models should be more flexible and let them specify which automatic links should be created, as well as the level of granularity to use. Our first implementation lacked this flexibility, as all the links identified in Section 4 are implemented compulsorily.

The development of the prototype also allowed us to evaluate the suitability of the hypertext system used for this application. With regards to the browsing subsystem, we felt that several features that could improve or simplify the searches are missing in Hyperties. These features were usually related to specificities of the minutes hypertext database. The possibility of specifying composite nodes, using typed links as described in [4] would allow users to see, when selecting the general node of a minutes, a continuous flow of text very close to the linear paper version, with no need of navigating through the minutes. Another useful feature would be the possibility of visualising the whole process from any one of its components. This could be done in two levels: by showing first a map of articles and then allowing the user to unfold any article.

The knowledge embedded within the statechart machines that model the

hypertext database supports the authoring activity, as the construction of the statecharts follows a detailed analysis of the documents to be inserted into the database. The static modelling requires the identification of the relevant information in the documents, and also indicates possible ways of linking such information. Atomic units of information identifyed during this phase will later correspond to states in the statechart models created for each type of document, and later on will appear as nodes associated to instances of documents contained in the hypertext system.

Moreover, the dynamic model represented by the statecharts support the implementation of algorithms for answering many queries common to database systems. These would be mostly queries looking for quantifiers, such as: How many articles of a certain subject are there in the hypertext database? How many times a certain attribute value appears as a link in the database? The model also supports queries looking for the first appearance of a subject and so on. More sophisticated queries could look for how many functions are still incomplete or yet how many have reached a final state or are stopped on a certain state. The statechart models also provide a valuable tool for supporting the dynamic growing of the hypertext database, enabling the authomatic linking of new nodes with nodes already in the database.

We believe that integrating the statechart machine model proposed in Section 3 for controlling links into a hypertex system would make available an extended system with greater power and potentially better usability. This new hypertext machine would control all instances of active functions within the system and support the statechart-based specification of document life cycles. The specification process could make use of tools such as the STATEMATE system [16] and the graphical editor STATSIM [20]. The availability of such a hypertext system, whose interface could be similar to the prototype one described by Stotts and Furuta [12], would greatly simplify the implementation of dynamic hypertext databases such as the one discussed in Section 4.

6 Concluding Remarks

We have shown how an office information system was modelled using a finite state machine based approach, and how this model allowed us to design a series of links in a hypertext database. We believe that the process of

performing searches in this database is close to the mental model employed by office workers to approach the activity of searching through a set of text documents without a clear classification apart from a sequence by filing date and some vague clues. Now the database is organised in a way that naturally facilitates the search.

The proposed model offers a mental model for users to perform searches that reduce the problems of disorientation they can experiment when searching a hypertext database. We believe that a wide class of office information systems based on the storage and retrieval of documents can benefit from hypertext organised databases, such as the formal face-to-face meetings application described in this paper.

Moreover, the statecharts provided a graphical way of visualising the flow of the documents within the system which proved very useful for the understanding of the model. They were also essential for automatically establishing the links within the database in a way that fully supports its growth, and were also used to impose integrity restrictions in the input of information. During the development of the prototype it was also verified that the dynamic models realised through statecharts allow the detection of inconsistencies in the static models and vice-versa, depending on which of them is developed first. This is a novel application of statecharts, in a different domain from the one they were originally conceived for.

The availability of on-line information about the minutes has also changed the behaviour of some users. Members of the Council had to await for the minutes to be released, and would approve it (or not) at the beginning of the next session. Now, they can look at the on-line database and print the minutes only if necessary; they can also browse through different paths when a topic is related to some topic appearing previously. Additionally, the summons can now be distributed by electronic mail, and a range of new possibilities for new office and computer technology exploitation is open.

References

[1] HARDMAN, L. Evaluating the Usability of the Glasgow Online Hypertext. *Hypermedia*, 1(1) 1989, 34-63.

[2] SHNEIDERMAN, B.; BRETHAUER, D.; PLAISANT, C.; POTTER, R.. Evaluating Three Museum Installations of a Hypertext System. *Journal of the American Society for Information Science*, 40(3) 1989, 172-82.

[3] CONKLIN, J. Hypertext: An Introduction and Survey. *Computer*, 2(9) 1987, 17-41.

[4] SOBIESIAK, R.; MYLOPOULOS, J. A Conceptual Modelling Approach to Authoring-in-the-Large for Hypertext Documents. *In:* Proceedings of the Conference on Organizational Computing Systems, Atlanta: ACM Press, 1991, (SIGOIS Bulletin, 12(2,3)), 225-39.

[5] FURUTA, R.; PLAISANT, C.; SCHNEIDERMAN, B. A Spectrum of Automatic Hypertext Constructions. *Hypermedia*, 1(2) 1989, 179-95.

[6] GLUSHKO, R. J. Transforming Text into Hypertext for a Compact Disc Encyclopedia. *In:* Proceedings of the Computer Human-Interface Conference, New York: ACM Press, 1989, 293-98.

[7] NUNN, D.; LEGGET, J.; BOYLE, C.; HICKS, D. *The REXX Project: A Case Study of Automatic Hypertext Construction.* Texas: A&M University Press, 1988 (Technical Report, Department of Computer Science).

[8] RADA, R. Converting a Textbook to Hypertext. *ACM Transactions on Information Systems*, 10(3) 1992, 294-315.

[9] WALKER, J.H. Support Document Development with Concordia. *IEEE Computer*, 23(1) 1988, 48-60.

[10] COGNETICS CORPORATION Hyperties User's Guide v. 3.0. Princeton Junction, 1991.

[11] NIELSEN, J. Hypertext & Hypermedia. San Diego: Academic Press, 1990.

[12] STOTTS, P. D.; FURUTA, R. Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. *ACM Transactions on Office Information Systems*, 7(1) 1989, 2-39.

[13] ZHENG, Y.; PONG, M. Using Statecharts to Model Hypertext. *In:* Proceedings of the European Conference on Hypertext Technology, Milano: ACM Press, 1992, 242-50.

[14] SUTHERLAND, J. OADM: *An Office Analysis and Diagnosis Methodology.* Cambridge: MIT Press, 1983. (MSc. Dissertation, MIT/LCS/TR-290).

[15] BOOCH, G. Object Oriented Design with Applications. Redwood City: The Benjamin Cummings Publishing Co., 1991.

[16] HAREL, D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8, 1987, 231-74.

[17] HALL, P. A.; PAPADOPOULOS, S. Hypertext Systems and Applications. *Information and Software Technology*, 32(7) 1990, 477-90.

[18] STEFIK, M.; FOSTER, G.; BOBROW, D.G.; KAHN, S.L. SUCHMAN, L. Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. *Communications of the ACM*, 30(1) January 1987, 32-47.

[19] SANTOS, G. P. B. dos *A Hypertext System to Support Formal Face-to-face Meetings.* São Carlos: ICMSC Press, 1992 (M.Sc. Dissertation, in Portuguese).

[20] MASIERO, P. C.; FORTES, R. P. M.; BATISTA, J. Editing and Simulation of the Behavioural Aspects of Real-Time Systems. *In:* Proceedings of the Brazilian Computer Society Conference, Santos: SBC Press, 1991, 45-61 (in Portuguese).

N O T A S D O I C M S C

Serie : Computacao

Nº 005/94 - NICOLETTI, M.C; MONARD, M.C. - Limiting the background knowledge in inductive logic programming

Nº 004/93 - NICOLETTI, M.C.; MONARD, M.C. - Learning horn clauses using the ILP system GOLEM

Nº 003/93 - ARENALES, M.N.; MORABITO, R.N. - An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems

Nº 002/93 - NICOLETTI, M.C.; MONARD, M.C. - Herbrand interpretation model and least within the framework of logic programming

Nº 001/93 - MORABITO, R.; ARENALES, M.N. - An and/or graph approach to the container loading problem