J. M Bishop
24/3/91

South African Computer Journal
Number 4
March 1991

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 4
Maart 1991

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083

Assistant Editor: Information Systems
Dr Peter Lay
P. O. Box 2142
Windmeul 7630

Subscriptions

	Annual	Single copy
Southern Africa:	R32-00	R8-00
Elsewhere:	$32-00	$8-00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Computer Science Department, University of Pretoria, Pretoria 0002
Phone: (012) 420-2843 Fax: (int) + (012) + 43-6454

Sigma Press

Guest Editorial

Does Today's Industry Need Qualified Computer Scientists?

This guest editorial consists of two contrasting views on the value to industry of a professional degree in computer science. Both authors, one local and one from Germany, are managing directors of well-respected software houses. (Editor)

Viewpoint I

Hans G Steiner
MBP Software and Systems GMBH
Semerteichestrasse 47-49
D4600 Dortmund 1

I would like to begin by recounting from my student days a story that I consider to be relevant. While attending a career forum for computer scientists, mathematicians and physicists, the personnel officer from IBM Germany was asked if he would consider taking on mathematicians. The gist of his answer was as follows: "Of course I must admit that I could just as well give the mathematician's job to a theologian. What is important is the ability to think logically. It is only there, on the job, that he learns how to become productive for us."

This episode occurred 14 years ago at a time when graduating mathematicians did not necessarily learn programming and when computer scientists were few and far between. The situation has improved immensely since then. Mechanical engineers, electrical engineers and physicists, all with programming knowledge, have for the most part taken over many programming jobs. This shows industry that, as time goes by, the answer to the opening question is becoming an ever-louder and more frequent "NO".

I support this opinion and in the remainder of this essay I will expand on my reasons, as well as highlight some exceptions.

An employee who is recruited directly from a university should possess the following four capabilities:
1. An ability to think logically: One of the basic requirements in our business is the ability to recognise, analyze, structure, break down and solve a problem as well as to fully synthesize the solution. The important thing is to break down the problem in such a way that the individual components can feasibly be solved. This is what distinguishes an engineer/scientist from an arts scholar. The latter usually concentrates on the complete problem and tends to settle for a contentious, complex and partially non-feasible solution. In our business, it is not enough to merely ask the "right" questions.

This ability to think logically may be accentuated in computer science; however engineers/scientists will generally possess the ability to an equal extent.

2. Programming skills: Our employees' prime tool of the trade is their ability to encode solutions to problems. Ideally, this ability ought to be held as abstract as possible. In other words, the further away from the "bit", the better. FORTRAN programmers who, for example, concentrate on the multiple use of memory space of all variables will never be successful programmers in an object-oriented programming language.

The difference can be seen, even in today's universities. For example, one only has to read a PROLOG program from a student who learned PASCAL in his first semester and PROLOG in his fifth. On average, this is always a "PASCAL program in PROLOG". The various possibilities offered by a predicate calculus language are only recognised and used by the best students. Again, we do not need the average computer science scholar who has spent between six and eight years writing complicated PASCAL programs, but rather the "thinker" with basic programming knowledge who is capable of abstracting the task. Once again, the ability is independent of faculty.

3. Teamwork skills: Working successfully in a team

This SACJ issue is sponsored by
Department of Computer Science
Rhodes University

requires assertiveness, tolerance, stability and one's own ideas. Very few problems have solutions that can be managed by one person successfully in the allocated time. Out of 700 employees, we can only afford approximately five "lone warriors" who are, in turn, the leading specialists in a wide field. They have a strategic vision which we follow. All remaining employees are evaluated, for better or worse, on their team performance. Some people have an in-built ability to work in teams. A few universities - unfortunately not enough - encourage this team-thinking. Again we see that the ability is independent of university faculty.

4. *Motivation:* The ability to enjoy one's particular job is a major driving force in every employee. Whereas in the sixties everything had to be "bigger, faster and better" and in the eighties "things had to be meaningful to society", the theme for the nineties is self-realization. Those companies who succeed in incorporating different employees (ie employees with different driving forces) into the company culture and who motivate each employee optimally will be successful in the nineties and beyond. There are huge productivity gains to be had from motivating employees. Compared with this, the possibilities offered by CASE tools pale into insignificance.

One basic requirement is thus the recruitment of a self-motivated employee who should at no stage become demotivated, whether it be by company culture, superiors or working conditions.

Again, this is not linked to a specific university faculty and is independent of know-how.

As none of these four capabilities are necessarily restricted to studies in computer science, the technical/-scientific background of new employees who are being recruited is largely irrelevant.

I would now like to point out a few exceptions which might give a computer scientist the upper hand in an interview. I refer exclusively to our own company and our specific company tasks.

1. Porting our COBOL Compiler onto the latest UNIX machine from the manufacturer XY. Knowledge of the UNIX operating systems could be very valuable and enable the new employee to rapidly become productive.
2. Programming the 37th interface (special customer request) for our ISDN card. Knowledge of interface protocols or experience with protocol conversions would be very useful and could be a decisive factor. Such specialized knowledge is usually very rare.
3. Adapting our integrated office automation system to the 17th foreign language. The employee must command the language perfectly. Simply outsourcing the translation would mean that this language version could not be maintained or supported. From this example one can see that specialized knowledge not only refers to knowledge gained from computer science studies.

In the product business, it sometimes happens that computer scientists with specialized knowledge are sought. (This is almost impossible in the project business, due to the variety of tasks to be performed.) However such a "knowledge" advantage over others usually only lasts about a year. After that, the achievements of two different employees (one with specialized knowledge and the other without) tends to even out.

Most applicants who start out do not know our products, as the flow of employees in this industry is almost always from manufacturer to user. Hardware and software manufacturers often lose their products specialist to the products' users. Seldom do employees change in the other direction.

In my opinion, universities can learn two things from this essay:

1. Studies in computer science give basic knowledge that can be used in various jobs. The student should however be careful not to place all his eggs in one basket.
2. Teamwork should be encouraged more. Time allows for very few geniuses, acting as 'lone warriors', to initiate progress in our society.

I have taken the liberty of basing my interpretation and answer to the opening question on my own judgement and experiences. I would be grateful for other opinions and experiences on this topic.

I would like to conclude by expressing my gratitude for having had this opportunity to express my views.

Viewpoint II

Pierre Visser
Grinaker Informatics, P.O.Box 29818, Sunnyside, 0132

The title question currently generates as many viewpoints as a counterpart question: "What is the correct curriculum for a computer science qualification?" Such questions stem from the many and diverse requirements expected to be fulfilled by the still developing applied science. A basic assumption of this editorial is that we need to have an explosion and consolidation in computer science theory. Only after this has occurred will a more general consensus of opinion exist - as is the case in other matured sciences.

An argument is presented here for the current approach of striving towards a balance between immediate industry needs and long term perceived theoretical requirements of industry, even though the balance, as viewed from either side, will always be imperfect.

Industry can, of course, do without qualified computer scientists - that is how it was established. Dedicated mathematicians, physicists, engineers and other scientists will, as in the past, continue to effect improvements. However, as one of those scientists from the

early days, it is difficult for me to understand why one would choose to continue this way.

A computer science qualification is viewed here as a university education (4 years) into theory that is not obtainable otherwise. By definition, therefore, a qualified computer scientist is not trained to conform to specific job requirements. Rather, the computer scientist will possess knowledge that will serve him long past the present day's computing technology.

Whether industry needs qualified computer scientists depends on two issues. Firstly, can an education be provided for computing technology that will serve as a foundation for the student's next 45 years in industry; and secondly, can industry build upon this foundation to created wealth more effectively than without qualified computer scientists.

It is widely accepted that, in broad terms, the teaching of fundamental theory will serve the first purpose. However, what subject matter to include from the wealth of mathematics, physics, OR, and from computing fields such as networking, operating systems and others, remains the illusive issue. Universities can merely strive to select the right mix for the perceived future needs of industry. This requires insight into the evolution of computing technology. I will later discuss such insight as a basic requirement for a qualified computer scientist.

What is important in teaching is to focus on fundamental theory. Just as the natural science student needs to breed fruit flies in order to gain insight into the dynamics of inheritance, so too the computer science student needs to develop software. The purpose should be to create understanding and insight into fundamental theory, and, just as in the case of the breeder of fruit flies, the software developed should never be measured against efficiency requirements from industry.

The second issue is whether industry can build on this theoretical foundation to create wealth.

A depth of insight into computing technology, more so than with other training, can be identified as the focus of the potential value of a qualified computer scientist to industry. Three areas which require such insight are discussed below, namely organisation, product definition and the application of new computing technology in industry.

Computing products form an integral part of an organisation, and represent a significant capital investment aimed at increasing efficiency. These products are incorporated in an evolutionary way to match changing organisational requirements with improving product capabilities. Decisions to use products determine the long term efficiency and cost-effective replacement. Such decisions require insight into computing technology and its evolution. A qualified computer scientist can improve such decisions only if he gains enough insight into computing technology as well as its interaction with business through years of practice.

The success of products in some areas is dependent on market requirements which depend on computing technology and its evolution. The correct definition of characteristics of products that interface to computers is such an example. Insight into computing technology is able to create the versatility, simplicity or other improved selling features which can open new market segments.

The third area where insight into computing technology plays an important role is in the application of new computing technology (or a new trend) in an organisation. Examples include the introductory period for networking, DBMS-technology, distributed processing and document image processing. In areas such as these, the newly qualified computer scientist can be applied effectively and at the same time build up insight through experience which he will require for the other areas of organisational and product decisions mentioned above.

A major dilemma in the continuous development of insight into computing technology by qualified computer scientists is their correct application in industry. The identification of the opportunities within the three areas discussed above, requires insight into computing technology itself. Winning companies that depend on computing technology have this ability. In such companies the insight of the qualified computer scientist into computing technology as well as its contribution to the business is constantly stimulated, turning the qualified computer scientist into a valuable company resource.

What has been neglected in this whole discussion is the role of the "technician" and of the casual user of computing technology. Such personnel are required to implement selected computing technology of the day efficiently, whether in accounting, chemical engineering or other specialised disciplines. Their role and place is unquestioned. However, it cannot be expected of them to evaluate the potential of new computing technology, formulate algorithms from fundamental theory or any such decisions which require insight built upon a sound theoretical knowledge of the field.

The final aspect in answering the opening question is whether the qualified computer scientist can outperform other professionals who build up their own experience in computing technology. Many examples could be cited of improvement brought about by non-computer scientists in the past. However, these individuals formed part of the bootstrapping for computer science theory and education. We should have faith in this bootstrapping of computer science qualifications, because computing technology will increasingly diversify into many directions of specialisation in years to come, each requiring a body of fundamental theory.

This complexity cannot be left to a casual development of insight - industry requires qualified computer scientists to experience interaction with business objectives in order to cope successfully with future computing technology.

Hypertext for Browsing in Computer Aided Learning[1]

J Barrow
Department of Computer Science and Information Systems, Unisa, P.O.Box 392 Pretoria

Abstract

Hypertext is presented as a database accessing mechanism well-suited to computer aided learning applications. On the basis of informal experiments with prototype software, some benefits of structured hypertext are explored. Multiple dynamic hierarchies model and filter information structure and content through various visualisations and offer a browsing framework, navigational support and orientation cues while retaining conceptual simplicity and incorporating web-learning principles.
Keywords: *Browsing, CAL, discovery learning, structured hypertext*
Computing Review Categories: *H.2.2, H.3.3, K.3.1*

Received January 1990, Accepted October 1990.

Introduction

The ability of computers to store and retrieve large amounts of information would appear to be well-suited to computer aided learning (CAL) situations [22, 13]. Particularly in less formal subjects such as history, geography or biology, learners could use a suitable database system both to browse generally and to view particular pertinent information.

Making this scenario viable requires a computer-human interface through which computer illiterate users (the learners) can readily retrieve the relevant sections of the database. Historically, database retrieval has been through keyword search using Boolean connectives such as AND and OR. This approach requires considerable skill in identifying suitable keywords and possibly many aliases, in formulating these in a suitable logical expression, and in broadening or narrowing the search should the initial query retrieve too little or too much information. Search performed in this way is also a black box process, and provides little help to the user either in initial query formulation or in subsequent refinement.

A database accessing technique like this is not well suited to CAL systems. In one study involving 9 to 11 year olds [21], simple database querying often required several formal paper-based stages and adult intervention before the required information was retrieved. Even several of the teachers involved could not successfully master the analytic aspects of more complex queries. These difficulties with query formulation are consistent with experiences in other areas (see, for example, [3]), and have occasioned much debate within the field of information retrieval. Even when enquiries are skilfully formulated, there tends in practice to be an inverse relationship between *recall* (that is, hit rate or proportion of relevant material retrieved) and *precision* (exclusion of irrelevant material) ([10] & [18]).

While the arrival of microcomputers led originally to the expectation that databases would find significant application in schools [9], an assessment of computer use in the classroom suggests that this has not materialised [5]. While the reasons for this were not investigated during this assessment, the research mentioned above suggests that the complexity of query formulation could be a contributing factor.

An alternative interface, which operates intuitively and at a more conceptual level (as opposed to a keyword level), is *hypertext*.

This article explores one particular approach to providing such an alternative. After providing a brief background to hypertext, it describes a simple hypertext prototype built to explore free associative linking. Informal evaluation indicated a number of difficulties which were also noted in the hypertext literature. This leads to a discussion of limiting hypertext links to hierarchies only, and to the description of a second prototype built to explore hierarchical hypertext.

This approach differs from many CAL systems in that it makes no attempt to provide active teaching. Rather, the intention is to provide visualisation of the underlying information structure and content, fast yet precise navigation over this information, orientation cues, filtering of extraneous information and an explicit set of structures to guide the learner and to offer a starting point for the learner's own internal knowledge representation. The goal is to produce an environment that facilitates both discovery-based learning and focused information retrieval while remaining conceptually simple.

[1] Based on a paper presented at the V[th] Southern African Computer Symposium, Johannesburg, December 1989.

Hypertext

A simplified view of hypertext is as an "electronic encyclopaedia". An encyclopaedia consists of many separate entries, each dealing with a specific topic. Along with each entry is a list of "See also" references, pointing the way to related entries.

A computer can readily be used to develop this concept of having nodes with links to associated information. While browsing through the system, the learner follows links which are important to him or her at that particular time, thus accessing the database in a very convenient and user-friendly way that closely resembles the associative manner in which humans often think. This has become known as hypertext.

Expressed less analogically, hypertext can be viewed as:

- a collection of information items in a database, with
- free, non-linear linking between these items, and
- window(s) through which to navigate across the links and view the items.

The links within this network can be of various types. The "See also"-type links relate associated information in different nodes. Other links may amplify the current context, allowing a reader to consult a glossary or to insert an annotation.

From the learner's perspective, there is no longer the distinctly separate stage of formulating a query - he or she simply follows relevant links visible on the screen, actively guiding the search process at each step. This potential benefit of hypertext as an alternative query mechanism in computer based training is also noted in [4].

Hypertext is a highly adaptable medium. A learner can choose the viewing path best suited to their needs from the rich set of associations available. For the benefit of learners with a poor background in a particular area, supportive material and glossaries can be offered without interfering with the flow of the main document. Learners can browse generally, following interesting associations and working both with the information content of individual nodes and with the structuring of this knowledge as represented by the links between the nodes. Alternatively, a learner can search very specifically, excluding all but the most relevant information. With the windowing facilities, learners can juxtapose different items to simplify comparisons.

Since this is a computer-based technology, the information base can be modified and updated easily, either by the learner as part of the learning process or by the teacher to suit changing conditions. Because the learner's model of the system is convenient and familiar, hypertext environments are easy to learn and to use in contrast to formal databases (as discussed above).

SuperText version 1

To try to assess some of these possibilities, a small hypertext system (called SuperText) was built at Unisa during 1988 [2]. Because it was intended for school use from the outset, the system scope was limited to low-cost technology, thus excluding approaches such as HyperCard and Guide, and to small, well-defined and intensively used databases.

To evaluate the concept, a portion of a Std 9 history text-book [12] was loaded onto the prototype software. The resulting system was then shown informally to a number of educators including one of the text-book's co-authors. The general feedback was that the concept held definite possibilities, but that several factors needed investigation.

The first was that hypertext makes it very easy to digress away from the original line of enquiry, and to change direction more and more with each successive link. If one is browsing generally, this can be an interesting and useful process, and is a useful contrast to more prescriptive kinds of CAL. But if one has a specific viewing goal, this digression can lead to disorientation and frustration.

To help learners re-orient themselves, a path retrace facility was introduced into SuperText. It is thus possible to move backwards and forwards along the viewing path at any time to re-establish the context. To help learners sustain a number of different viewing paths during one session, a bookmarking facility was created. This allows a learner to scan the available information and mark relevant areas before starting in-depth viewing, or to mark nodes for later perusal. These two facilities amount to the introduction of a new kind of linking that is created dynamically in response to each individual learner.

The second factor to consider was that reading hypertext is more demanding than reading conventional linear text. It is not just that reading from a computer screen is more demanding than reading from paper. The learner must now both assimilate the basic information, and continually select which of the available links to follow. Before selecting a link, the learner must answer, though probably not at a conscious level, the questions of whether the relationship represented by the link is relevant to his or her current learning goals and how this would affect his or her internal knowledge representations. However, this cannot happen without some knowledge of what the target node contains. While it is easy to indicate the presence of a link on a computer screen, it is much more difficult to indicate the nature of the target node. Consequently, learners are likely both to follow irrelevant links (low precision) and to miss important links (low recall).

To facilitate the selection of relevant links, thus improving both precision and recall, a fast link previewing facility was developed for SuperText.

Whenever a link is highlighted, the first few lines of the destination node are displayed in the preview window. The learner can now gauge the type of information a link points to before actually selecting it. This reduces the frustration of repeatedly selecting inappropriate links and then needing to backtrack and the concern over missing important information when ignoring links.

The third difficulty lay in trying to determine whether specific information is available on the system and in how to access it if it is, since associative linking is often not self-evident. Most hypertext systems address this problem outside the hypertext structure through some form of the keyword search already discussed.

While these three factors were identified informally, they have all been confirmed by other workers in the field [6, 1, 8, 19]. Thus a path retrace facility is virtually a standard hypertext feature. Guided tours are used to introduce learners to a hyperdocument, and manually constructed graphical browsers can portray some sense of the overall linking. Attaching semantics to a link has been approached by adopting a "rhetoric of hypertext" [15] and by a frame structure resulting in node-link-node triples [17].

However, there is another approach to these problems. All three are, at least in part, a consequence of the very free nature of the associative linking. Possibly, if one were to provide some inherent structure to the hyperdocument, the problems of digression, of increased cognitive load, and of locating required information could be reduced. Surprisingly, this is a relatively unexplored area.

Hierarchically Structured Hypertext

Returning briefly to analogy, books are hierarchically structured by means of chapters, main headings, minor headings, subheadings, and so on. This structure is reflected in the table of contents, which presents both the overall structure of the book, and a useful means for locating likely areas of interest. Similarly, a library card index is housed in cabinets consisting of drawers containing ordered subgroupings of index cards. A card index also allows the same information to be indexed in more than one way, usually by author and by subject.

Transferring this analogy to hypertext, the initial hypertext model can be extended to include multiple structured views of the information items in the database rather than a single "table of contents" view. As an added refinement, these views can be made expandable or contractible to provide a dynamic display either showing or hiding lower levels of the hierarchies (referred to as "zooming" or "telescoping")[1].

What benefits do the provision of multiple views through dynamic hierarchical structuring offer? When using the system, the learner would first select a suitable view of the information base. For example, in a school history system, the same information may be structured differently into views emphasising chronology, particular themes and cause-effect relationships.

As a first benefit, zooming provides a concretisation of web teaching and web learning principles. According to [16], effective teaching results from first presenting a framework for relating materials to each other (the web), corresponding to a zoomed out view, before presenting the detail, seen here by zooming in. Over time, the same information will be encountered in different contexts (different hierarchies), so enhancing integration of new information into prior knowledge.

Secondly, by collapsing the chosen hierarchy, the learner can rapidly navigate over large distances to find the required general grouping of nodes, and can then successively expand the hierarchy to reach the detailed selection(s). Thus, fast, coarse positioning through to fine, detailed positioning is provided by zooming within a given structure. In contrast, free associative hypertext captures no such abstraction information, and higher level representations cannot be generated automatically in a meaningful way [23].

The third factor to consider is user disorientation. Free linking almost invites major digressions from the initial line of enquiry, and, as mentioned, there is no easy way to zoom visual images of a network. In contrast, a hierarchy provides boundaries within which to browse, and can be readily visualised and manipulated.

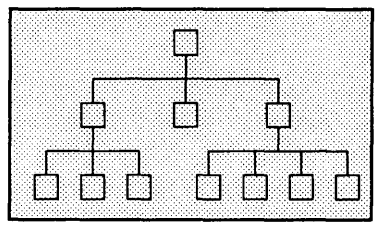
Figure 1

Hierarchies are commonly represented as trees, either vertically (as in figure 1) or horizontally (as in figure 2), but can be represented in other ways too [7,14].

Part-whole relationships are clearly displayed through a nested subset representation, where lower hierarchical levels are recursively nested within higher ones, as in figure 3.

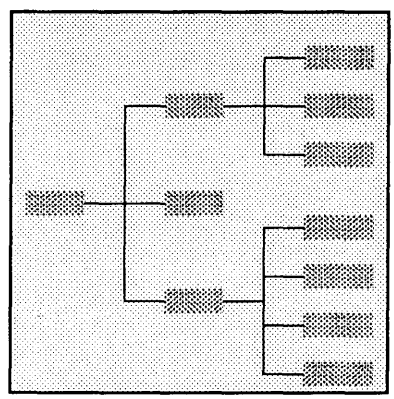
Figure 2

Alternatively, a particular subtree can be shown through a "mind-map" or spider diagram, where descendants are recursively clustered around their parent, as in figure 4.

Another possibility is to assign each level of the hierarchy to a separate window, with coordinated

scrolling between windows (see fig 9 in [6]). For text with basically linear characteristics, the top-level siblings can be assigned to successive cascaded windows, thus indicating "where" the learner is in a particular view and simulating the physical position in a book (see figure 5).

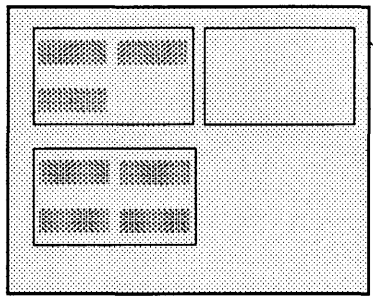

Figure 3

The possibility of generating these different representations automatically by computer introduces a fourth benefit of dynamic hierarchies. It offers the learner an external representation of the knowledge structure. The learner now has a starting point for creating his or her own internal knowledge structures (in whatever form they may be).

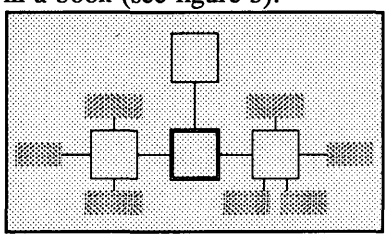

Figure 4

These diagrams are generally not very compact, and a number of restrictions arise when they are applied to a small computer screen. In contrast to a very flat database structure, the number of children per parent should be limited to reduce the fan-out. The number of levels that can be shown simultaneously is limited, and the title of each node must be concise and descriptive.

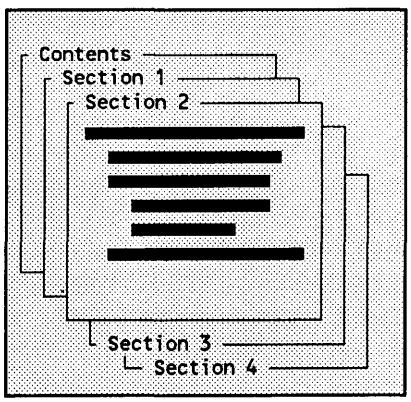

Figure 5

To achieve greater information density on the screen, the text-based convention of showing hierarchy by indentation level can be incorporated[2], as illustrated by figure 6

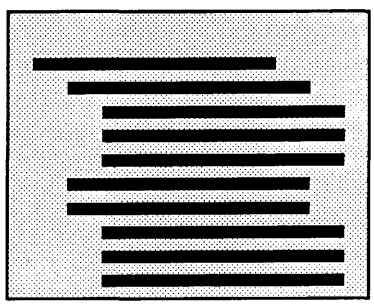

Figure 6

To make structure quite explicit on a dynamic display such as a computer screen, tree diagramming or nesting can be combined with indentation levels, as in figure 7.

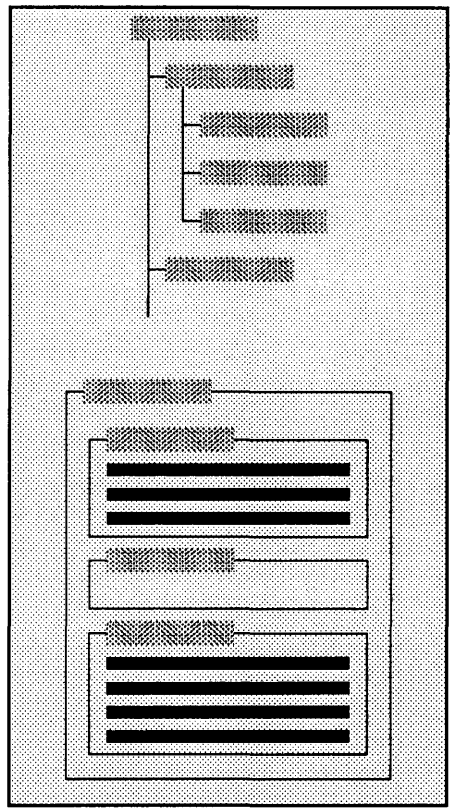

Figure 7

Used appropriately, these structures offer the learner a clear visual reminder of the context of the current text item and of the structure of the current view, and so help combat disorientation. The hierarchy is a familiar and highly pervasive model that makes a clear statement about relationships between items of information and that has an intuitive filtering characteristic. The learner's mental load is reduced since it is necessary to select the next item consciously only when none of the existing views meet the learner's requirements.

SuperText version 2

Following from the considerations given in the previous section, the application of hierarchies within the CAL context seemed to justify further investigation. Initially, I adopted the common approach [19] of setting up hierarchies explicitly through the associative links available under the first SuperText prototype. In effect, this created contents pages and information pages. However the resulting structures are static, and the system itself has no knowledge of their existence. Consequently, expanding or contracting the hierarchy is not possible, and visualisation strategies such as those discussed above cannot be employed. Many of the possible benefits of this structuring were not realised, and so a second version of SuperText was undertaken specifically to evaluate dynamic hierarchies.

Concern was felt over the effects of hierarchical structuring on the authoring process. Writing any document is an evolutionary process, often starting with indefinite and disconnected ideas which are subsequently expanded and structured in an iterative way. Consequently, the main problems anticipated with an approach as apparently rigid as a hierarchy were that the author (ie the teacher) would be forced into premature structuring or into single-dimensional

structuring [11]. It was also realised that the nature of the hyperdocument should not be static, but should be able to reflect changes resulting as concepts develop during authoring or as new information is added.

To accommodate these factors, SuperText treats the actual nodes within the database as separate objects independent of their hierarchical relationships. Multiple hierarchical structures linking these nodes can thus be created, and each node can be referred to repeatedly and in different views. One of these views is designated as a work area, where nodes can be created in an *ad hoc* fashion and stored until the required hierarchies have been determined.

Currently, visualisation of the hierarchies is through indentation level, coupled to either numbering or tree diagramming. Structure editing has been given particular attention with functions such as subtree pruning and grafting within or between views, and subtree promotion or demotion in addition to functions relating to individual nodes alone, such as creation, deletion and splitting being provided. An interesting side-effect is that sub-trees are effectively simple aggregations or compositions, with some operations applying to an entire aggregation rather than to an individual node [11].

The opportunity has also been taken to introduce two orientation features to the hierarchy display. If a particular node has already been viewed (whether in the current hierarchy or another one), a "footprint" is placed alongside its entry. A "scope-bar" indicates what proportion of the available nodes have been viewed. The learner consequently has continuous feedback on what has been seen and on how much remains.

The text editor supports standard functions such as comprehensive cursor control, block functions, and word-wrapping. The structured document can be printed on a standard printer. The system uses the single floppy drive IBM-PC architecture as a minimum.

Applying these structures

It is anticipated that the approach described here will apply to a particular subset of hypertext applications and of CAL subjects. The subject area should be clearly demarcated, and should lend itself to categorisation. It should be possible to "chunk" the information in small concise units with a clear descriptive title.

The information base should be relatively self-contained, should be used extensively, and, once created, should only change slowly. In common with other CAL systems, significant authoring effort is required. Thus courseware would be developed centrally by a team for general use as part of an overall policy rather than by a single teacher for local use.

In accordance with web-learning principles, the learner would generally make several passes through the material. Initial viewing would be at a high level for familiarisation with the hierarchy "zoomed out". Subsequent passes would be both in more detail, to consolidate and expand the earlier reading, and of different hierarchies, to extend knowledge and enhance integration.

Specific details such as the use of different visualisations would depend on the application. For a school biology system, the nested display may be particularly appropriate to emphasise part/whole relationships (petals are part of a flower are part of a plant). This could be fixed at authoring time, simplifying the interface for the learner. Conversely, a more computer-literate user may well prefer to select the most useful display type at viewing time.

Conclusion

In this article, I have argued that difficulties with keyword-oriented query formulation hamper the application of conventional databases as browsing media in education. Unstructured hypertext offers a more intuitive interface with a straightforward query mechanism through link following.

However, informal tests with a prototype system suggested that unstructured hypertext may involve significant penalties in the form of learner disorientation and cognitive load. Thus an alternative, structuring hypertext through multiple dynamic hierarchies, is proposed. The possible benefits are the provision of an external cognitive model through different visualisations and filtering of the information structure and content, and of a framework offering browsing guidelines and orientation cues to the learner. The resultant user interface is conceptually simple, and offers fast yet precise navigation through the available information.

This approach can readily be extended to full hypermedia systems with large information bases and incorporating technologies such as CD-ROM and Videodiscs. The nodes themselves can be extended to include graphics, simulations, and video sequences.

Notes

1. Although this was initially a hypertext idea, it has fallen away in more recent hypertext implementations. It now lives on mainly in the "outliner" category of software [6, 7]. It has also been used as a browsing strategy as suggested here [20].

2. Paper-based systems can use heading size and style, but this is not possible with many computer displays, and is generally not as explicit as the other methods. A numbering scheme can also be used, possibly in conjunction with indentation levels.

References

[1] Acksyn R, McCracken D and Yoder E, [1988], KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations. *Communications of the ACM*, July, 820-835.

[2] Barrow J, [1989], Hypertext as an Educational Medium. *First Southern African Conference on Educational Technology*, HSRC, Pretoria, July.

[3] Campagnoni F and Ehrlich E, [1989], Information Retrieval using a Hypertext-based Help System. *12th ACMSIGIR Conference*, June.

[4] Carr C, [1988], Hypertext: A New Training Tool? *Educational Technology*, August, 7-11.

[5] Chan C, [1989], Computer Use in the Classroom - II. An Assessment of Using the Computer as a Tool and as Tutee. *Computers in Education*, **13**(3), 271-277.

[6] Conklin J, [1987], Hypertext: An Introduction and Survey. *Computer*, Sept, 17-41.

[7] Feiner S, [1988], Seeing the forest for the trees: Hierarchical display of hypertext structure. *Proceedings of the Conference on Office Information Systems (Palo Alto, Calif)*. ACM, New York, 205-212.

[8] Foss C, [1987], Effective Browsing in Hypertext Systems. *CeRCLe Technical Report No 41*, Centre for Research on Computers and Learning, University of Lancaster, England.

[9] Freeman D and Tagg W, [1985], Databases in the classroom, *Journal of Computer Assisted Learning*, **1**, 2-11.

[10] Gordon M and Kochen M, [1989], Recall-Precision Trade-Off: A Derivation. *Journal of the American Society for Information Science*, **40**(3), 145-151.

[11] Halasz F, [1988], Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, July, 836-852.

[12] Kallaway P (ed), [1986], *History Alive 9*. Shuter & Shooter, Pietermaritzburg.

[13] Knight P and Timmins G, [1986], Using databases in history teaching. *Journal of Computer Assisted Learning*, **2**, 93-101.

[14] Knuth D, [1973], *The Art of Computer Programming (2nd ed), Vol 1 Fundamental Algorithms*, Addison-Wesley, Reading, Mass.

[15] Landow G, [1988], Hypertext in Literary Education, Criticism, and Scholarship. *Computers and the Humanities*, **23**, 173-198.

[16] Norman D, [1973], Cognitive organisation and learning, *Tech Report ED 083 543, Center for Human Information Processing, Univ of Calif, San Diego*, reported in D Jonassen, [1989], *Hypertext/Hypermedia*, Educational Technology Publications, Englewood Cliffs.

[17] Rada R, [1990], Writing a Hypertext Book: The Role of a Semantic Net. *Unpublished manuscript*.

[18] Salton G, [1986], Another Look at Automatic Text-Retrieval Systems. *Communications of the ACM*, July, **29**(7), pp 648-656.

[19] Shneiderman B, [1989], Reflections on Authoring, Editing, and Managing Hypertext. In: Barrett, E (ed) [1989]. *The Society of Text*. MIT Press.

[20] Shneiderman B, Shafer P, Simon R and Weldon L, [1986], Display Strategies for Program Browsing: Concepts and Experiment. *IEEE Software*, May, 7-15.

[21] Spavold J, [1989], Children and databases: an analysis of data entry and query formulation. *Journal of Computer Assisted Learning*, **5**, 145-160.

[22] Stolurow L, [1969], Computer-assisted instruction, in H James (ed), *The Schools and the Challenge of Innovation*, Mc-Graw Hill, quoted in T O'Shea and J Self, [1983], *Learning and Teaching with Computers*, The Harvester Press, Sussex.

[23] Utting K and Yankelovich N, [1989], Context and Orientation in Hypermedia Networks, *ACM Transactions on Information Systems*, Jan, **7**(1), 58-84.

Acknowledgement

I wish to thank the journal's anonymous referees for drawing my attention to various shortcomings in the original version of this article.

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as a Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted **in triplicate** to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
- References should be listed at the end of the text **in alphabetic order** of the (first) author's surname, and should be cited in the text in square brackets. References should thus take the following form:

  [1] E Ashcroft and Z Manna, [1972], The translation of 'GOTO' programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

  [2] C Bohm and G Jacopini, [1966], Flow diagrams, Turing machines and languages with only two formation rules, *Comm. ACM*, **9**, 366-371.

  [3] S Ginsburg, [1966], *Mathematical theory of context free languages,* McGraw Hill, New York.

Manuscripts *accepted* for publication should comply with the above guidelines, and may be provided in one of the following formats:

- in a **typed form** (i.e. suitable for scanning);
- as an **ASCII file** on diskette; or
- as a **WordPerfect**, **TEX** or **LATEX** or file; or
- in **camera-ready** format.

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies. A styles file is available from the editor for Wordperfect, **TEX** or **LATEX** documents.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 4, March 1991
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 4, Maart 1991
ISSN 1015-7999

Contents