Vulcan: A Modified Viewdata System with Improved Indexing and Formal Hierarchy Definition Language

John A. Mariani and Douglas R. McGregor
Department of Computer Science, University of Strathclyde, Glasgow, UK

Conventional viewdata systems have two main drawbacks. First, they store information in a hierarchic structure which may be more naturally held as a network. As a result, a user may find himself forced to search through the structure in an illogical fashion. Further, the information provider, who is expected to supply not only the data but also its structure, is given very little aid. This paper describes the facilities of a modified viewdata system, Vulcan, which overcomes these problems, and its associated hierarchy specification language, VPREP. Information can be structured as a 'natural' network but is presented to the user as a logically consistent hierarchy. The VPREP language offers formal assistance for the specification, development and maintenance of the information network structure. The system described has been in use for several months.

INTRODUCTION

Viewdata systems—such as Britain's Prestel[1]—form one of the major man–computer interfaces in everyday use throughout the world.[2] These systems all share one factor in common—their use of a hierarchic information structure, which presents a menu-driven interface to the user. Information, held as 'pages' by these systems, can be accessed either as part of the hierarchic structure or directly via page numbers.

Information is located by following a path through the hierarchy, guided by routing and indexing information held on other pages. As the user works through the indexes, he gets closer and closer to the required information. The hierarchic structure can be thought of as a tree. The user starts at the root of the tree and traverses its branches towards the relevant information. However, if the information the user requires is scattered on different branches of the tree the user must repeat the search on each branch. Further complications can arise if the user is unsure of the index classification of the information required. For example, is a '*Motoring Which*' page under the 'Motoring', 'Magazine' or '*Which*' branches?

The Vulcan system has been designed to overcome this basic weakness of the hierarchic information structure used in existing viewdata systems. In effect, the Vulcan system allows the data to exist as a more natural network of pages (therefore allowing all 3 branches in our example to lead to the '*Motoring Which*' page) but superimposes a standard viewdata hierarchic structure over the network.

This allows information to be indexed under any number of branches and therefore increases the chances of the user finding the required page. It is possible to index pages under Prestel, but accessing such a page may require the user to jump to a completely different branch. Vulcan allows such page accessing to continue in a controlled descent of the imposed hierarchy. To ease navigation through the hierarchy, Vulcan uses structured page numbers (detailed below).

The emphasis of viewdata systems has always been on the simplicity of the system for the end-user—enabling him to grasp the commands easily and to become acquainted with the system quickly. However, the number of pages now available on systems such as Prestel is over a quarter of a million. The management of such a vast number of pages is as a horrendous task—even if it is left to the information providers to maintain their own branches of the overall database. Equally troublesome is the specification and development of the structure of the branches. To overcome these problems, Vulcan has an associated specification language called VPREP which provides a formal mechanism for the specification and development of page structures.

This paper is divided into four sections. In the next section, the use of structured page numbers is detailed, then the method of presenting the network of information as a logically consistent hierarchy is described. The implementation and use of the current system is then outlined. In the third section, the VPREP language, which provides a formal mechanism for the specification and development of page structures is presented. The fourth section lists a number of future developments and extensions for the overall system, before some concluding remarks are presented in the final section.

THE VULCAN SYSTEM

Structured page numbers

The Vulcan system emphasizes the hierarchic structure of the data by using structured page numbers which more closely reflect the information structure than simple page numbers. For example, '42.1.14' gives the user a better feel for his position in the information structure than '42114', and leaves the user in no possible doubt as to how he reached his present position. The standard formalism (42114) gives multiple possible paths (4.21.14, 42.11.4, 42.1.14, etc.). Simple operators allow the user to descend or ascend any hierarchy.

CCC–0010–4620/84/0027–0121 $03.00

The network hierarchy

Conceptually, there are three categories of pages in the Vulcan system.

1. A B-page, or 'base' page. This is, in the current implementation, a simple, individual text file which has its page number as its name. A B-page contains the data of interest to the user.
2. An I-page, or 'information' page, which is a pointer to a B-page.
3. An R-page, or 'routing' page. These provide a local index to the I-pages immediately below the user's current position. R-pages form the nodes in the index hierarchies.

Vulcan achieves its network effect by allowing one base page (a B-page) to be referenced from any hierarchy by an I-page. This therefore allows our network of B-pages to be presented to the user as a logically consistent hierarchy of I-pages.

For example, in the hierarchic branches described in Table 1 and Fig. 1, two 'base' pages, B-3268 and B-5962, contain data relating to theatre and cinema performances in London, respectively. These 'base' pages can be accessed via two different, separate branches of directory pages (R-pages). Thus a user can look up his desired page (say B-3268) using the 'Entertainments' branch starting at I-page 10 (or any subsequent node of that branch, such as 10.10). Thus far the system's user interface is exactly like the traditional one. However, he could also access the required information using the 'London' hierarchy starting at I-page 40 (or lower nodes such as 40.3).

Table 1

10	: Entertainments	40	: London
10.1	: Glasgow	40.1	: Dining Out
10.2	: Edinburgh	40.2	: Public Transport
⋮		40.3	: Entertainments
		⋮	
10.10	: London	40.9	: Parks and Gardens
10.10.1	: Theatres	40.3.1	: Theatres
10.10.2	: Cinemas	40.3.2	: Cinemas
⋮		⋮	
10.10.9	: Exhibitions	40.3.9	: Exhibitions

In effect, we have introduced a level of indirection between the logical page (I-page) as seen by the user and the actual page (B-page). This allows a many-to-one mapping between I and B pages, and hence many indexing branches can be mapped onto a network.

The benefits of such a structure, to the casual user, are as follows:

1. He can follow a meaningful path to an end-page, rather than start following one branch only to find he must then move to another page on another branch.
2. It is easier for a Vulcan user to have a better understanding of the topography of the database. For example, he could 'check out' the current cinema offerings in London and know that he need only go up one level and down another path to discover the theatre presentations. He never loses himself in a plethora of numerical series.

The set of user commands is given in Appendix 2, followed by an example of the use of the system in Appendix 3.

There are also implications for the information provider. As the network allows pages to occur in various indexing hierarchies, the problems of multiple classifications are eased. The '*Motoring Which*' page referred to earlier can now appear under the 'Motoring', 'Magazines' and '*Which*' hierarchies and any further suitable indices. In a hierarchy as we have described, consisting solely of real pages with duplications where required, an update to a page (a change of price for a car on a '*Motoring Which*' page, for example) must be performed wherever a page (and any duplicate) appears. If there is no mechanism which ensures that the person changing or updating the information is aware of all the duplicates, copies may be left unaltered. Under Vulcan, a change to any real page is naturally propagated to all I-pages.

Data structure and access mechanism of experimental system

The current experimental Vulcan system maintains its network of pages by storing the necessary information in a database management system (DBMS). The DBMS used is a locally developed system called Alpha,[3] which is a simple, sub-relational system which handles data

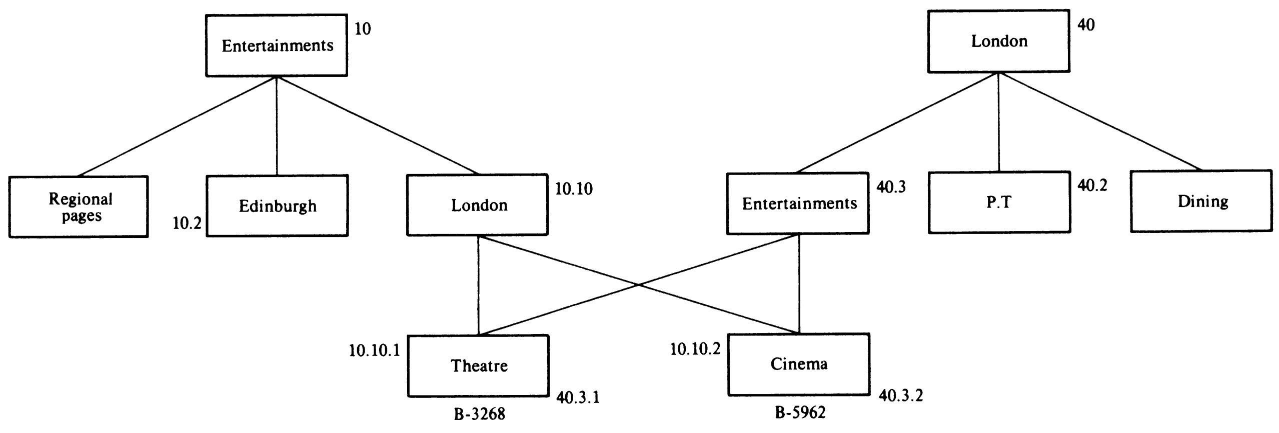

Figure 1

conceptually in the form of a table, with each line in the table a record or tuple, and each entry in the table a field.

A page in Alpha is identified by a record which has the format

(I-page number: B-page number: index-line)

When a request to view an I-page arrives, the database is scanned to find the corresponding B-page. If this search is successful, the B-page is displayed to the user.

An index-line is a brief, one line description of the I-page's content. Via Alpha, the user can receive an index listing (R-page) of all the pages below his current position. R-pages are assembled dynamically by scanning the Alpha database for all followers of the current page and retrieving the associated index lines, which are then sorted and displayed to the user. Using this method, R-pages of any depth can be generated, i.e. not just those immediately below the current page, but also the followers of the followers, etc. Therefore any alteration to the indexing information, be it the addition or deletion of pages, is immediately reflected in the R-pages.

In conventional viewdata systems, pages can be totally taken up by 'routing' information rather than information itself. In the Vulcan system, cases will naturally arise when only routing information (held in the R-pages) is applicable. When this occurs, we have three options for a redundant I/B page:

(i) it does not exist
(ii) it exists, but holds no information
(iii) it exists, and duplicates the R-page.

The second case is wasteful, and the third case re-introduces the problems solved (inflexibility in re-arranging the hierarchy, update problems in the routing pages) by the R-pages. I-pages must exist as necessary links in the network, but in Vulcan it need only exist as a 'page record', with a null B-page field. When the user requests such a non-existent I/B-page, the matching R-page can be displayed by default.

In commercial systems, these liberated I-pages could be used to carry advertisements which would presumably be connected with the current classification. In our previous example (shown in Table 1), in response to a request for page '40.1', an advertisement for a specific restaurant might appear. If the user requests a current index, Table 2 would appear.

Table 2

40.1.1:	Italian Restaurants
40.1.2:	Chinese Restaurants
40.1.3:	Indian Restaurants
⋮	
40.1.9:	Arabic Restaurants

System usage

The implementation of Vulcan was originally undertaken to allow the investigation of a typical menu driven interface to data, in order to allow comparisons with our other database work involving query languages. However, it outgrew this specification and came to its present use as a viewdata system.

As the system was mounted on a host operating system, it did not address any communication problems, leaving the underlying operating system to deal with these considerations. Freedom from such matters allowed us to concentrate on the man–machine interface aspects of the system. Vulcan has evolved over the past two years since its conception from a very simple file oriented approach and its investigative purpose to the system described in this paper.

Since its earliest implementation, Vulcan has been in use both as a departmental notice-board/magazine and, primarily, as part of the local AutoProg system.[4] The AutoProg system essentially maintains a library of user source modules, accessible by all users. Vulcan plays a very important role within the system by providing a catalogue of the available modules, therefore promoting the re-use of existing code. The version of Vulcan described in this paper helps alleviate the classification problems experienced by both the donating and receiving programmers connected with such catalogues.

A FORMAL LANGUAGE FOR THE SPECIFICATION AND DEVELOPMENT OF PAGE STRUCTURES

Introduction

Viewdata research has always been aimed at the end-user, with little assistance directed at the information provider. When a page is added to a viewdata structure, there is normally no assistance in placing the page, or assurance that the pages referenced by the new page do (or will) exist. In standard systems, there is a great deal of cross-referencing across branches. When a new page is added, previous cross-referencing can be forgotten, resulting in a page which is either over or under referenced.

We now outline the VPREP language, which offers formal assistance for the specification, development and maintenance of page structures.

The VPREP language

Perhaps the best way to introduce the very simple VPREP language is by means of an example. A full specification of the syntax of VPREP is given in Appendix 1. The most basic unit of the language is a page. The declaration of a page is shown below—we will use this example to highlight the constructions available within the language.

```
1  pagename bbc_radio;
2  indexline 'BBC radio programs'
3  [
4  big 'BBC Radio';
5  Programmes on BBC radio tonight—
6  Select 1 for radio 1 ⟨points. to bbc_radio_1⟩
7         2 for radio 2 ⟨points. to bbc_radio_2⟩
8         3 for radio 3 ⟨points. to bbc_radio_3⟩
9         4 for radio 4
10 ];
11 points. to bbc_radio_4;
12 pointed. at. by bbc_entertainments;
13 endpage bbc_radio;
```

Line 1 shows the *pagename* statement. This allows the user to give each page a symbolic name that he can refer to throughout the specification. Line 2 gives the index line that should be used to describe the page—this is applicable in the Vulcan system.

The opening square bracket on line 3 indicates the start of the data we wish to be presented to the user when he selects the page in question. However, VPREP statements can still appear within the data. A special example of this appears on line 4—the *big* statement is a formatting directive which specifies the string following it is to appear in large letters at the top of the page.

Lines 6, 7, and 8 show VPREP statements appearing within angled brackets. The *points. to* statement is followed by a list of one or more page names and indicates that the page currently being compiled points to the page listed. The statement can also appear after the closing square brackets (as shown on line 11) which indicates the end of the data. Allowing the *points. to* statement to appear within the data enables the user to place the appropriate *points. to* statement at a logical place within the data.

Line 11 shows the usability of the *points. to* statement outside the data area. Line 12 introduces the *pointed. at. by* statement which gives a list of pages that point at the current page. This is not strictly necessary but assists with structure checking at the end of the compilation process.

During the compilation process, a database is constructed which contains all the structural information gathered from the VPREP statements. Exhaustive analysis of this database is performed to ensure that

1. All declared pages (except the root page) are pointed at
2. All the pages specified as being *pointed. to* exist
3. The *pointed. at. by* information matches the *points. to* information.

In this way, the network structure is verified.

The next state in the compilation process is to generate the viewdata database in suitable format. This is system dependent and the current VPREP compiler is based on the Vulcan system. The compiler starts at the root page and gives each page one or more logically consistent page numbers. In this manner, the page specifier is liberated from the task of ensuring that each page is given a suitable number.

FUTURE DEVELOPMENTS

Symbolic path names

We are currently investigating techniques whereby meaningful symbolic names could be employed efficiently by users e.g. 'LONDON. ENTERTAINMENT. CINEMA' instead of '40.1.10' (or perhaps a mixture of both symbolic names and page numbers).

Page classification

In the current system, when a new page is added to the database, it is not automatically assigned to any location(s) within the network. It is then up to the system manager to decide the indices that could validly reference the page. Possible solutions could involve the provision of key-words (as in conventional retrieval systems) to aid the system manager directly or, if key-words were selected from a fixed collection of such words, the system could identify similar occurrences and bring prospective locations to the attention of the manager.

Supervision of the hierarchy

The structure of the data pages exists only in the DBMS. Therefore, to delete a page, a record referencing that page is removed from the database. To prevent the deletion of a hierarchy, no page record which represents a parent node can be erased.

Control of the hierarchy is achieved by a simple program interacting with the DBMS which allows addition and deletion of such page records. By repeated application of such insertion/deletion actions, complete subhierarchies can be added or repositioned in the overall structure.

Extensions to the VPREP language support system

The formal definition exists only as the information provider's specification. If the provider wishes to add, delete or move pages—or entire page structures—the only way is to edit the specification and recompile. This is undesirable in the context of a quarter of a million pages.

The VPREP system needs to be extended to allow online alterations to be made to the database while still dealing with it in terms of the initial specification. In addition, it might be useful to be able to generate a new specification from the current page structure, which an information provider can then take away and work on in terms of the VPREP language.

Analogous to conventional programming languages, the VPREP system should provide separate compilation of segments of the hierarchy. A tool, similar in function to a linker, could then link such segments into the overall hierarchy. This could allow individual information providers to produce their segments separately from others.

CONCLUDING REMARKS

In this paper, we have presented a model of a viewdata system which allows information to be structured in a network but presented to the user as a logically consistent hierarchy. With the additional support of structured page numbers, these factors combine to present the user with an easier navigational interface to the information.

The VPREP language provides a much needed formal definition of page structures for viewdata systems. It is also a practical aid to information providers, relieving them of tedious and error-prone tasks and alerting them to any structural errors.

The model considerably simplifies two main aspects of viewdata systems, the insertion and the finding of pages. Insertion has been simplified in that the information provider can supply any number of classification headings he feels suitable, and use a formal language to specify the structure of the pages. Similarly, the user has a greater chance of locating the page he requires as he has a larger number of routes to follow.

Acknowledgements

The authors wish to thank David Hutchison, Doug Shepherd and the referee for their helpful comments on earlier versions of this paper.

REFERENCES

1. S. Fedida, Viewdata: an interactive information service for the general public. *Communications Networks*, London, England, Sept. 1975 (Uxbridge, Middx., England: Online) pp. 261–282 (1975).
2. R. Woolfe, *Videotex—the New Television–Telephone Information Services*, Computer Science Series, Heyden and Sons Ltd. (1980).
3. J. A. Mariani, *Alpha—A Tabular Database Management System*, Internal Document, Dept. of Computer Science, University of Strathclyde.
4. D. R. McGregor and J. A. Mariani, AutoProg—a software development and maintenance system. *The IUCC Bulletin* **3** (1) (1981).

Received February 1983

APPENDIX 1

Syntax of the VPREP language

Key to syntax.

⟨*a*⟩ denotes a variable

b denotes a terminal symbol

⟨*x*⟩ : : = *y*⟨*z*⟩ means that the variable ⟨*x*⟩ is made up from the terminal *y* followed by the variable ⟨*z*⟩

⟨*x*⟩ : : = ⟨*a*⟩|⟨*b*⟩ means that ⟨*x*⟩ is made up from either ⟨*a*⟩ or ⟨*b*⟩

⟨*x*⟩ : : = ⟨*a*⟩|⟨null⟩ means that ⟨*x*⟩ is made up from either ⟨*a*⟩ or the special variable ⟨null⟩, which means ⟨*x*⟩ would be empty.

⟨*x*⟩ : : = *a*⟨*x*⟩|⟨null⟩ means that ⟨*x*⟩ is made up of no symbols (is empty) or any number of *a*s.

⟨string⟩ is a sequence of characters which do not form a terminal symbol.

Terminal symbols are delimited by one of the following: space or end-of-line.

```
⟨pages⟩ : : = ⟨page⟩ ⟨pages⟩
            |⟨page⟩ end
⟨page⟩ : : = ⟨pagehead⟩ ⟨datasection⟩ ⟨pageend⟩
                                           ⟨pagefinish⟩
⟨pagehead⟩ : : = pagehead ⟨string⟩; indexline ⟨string⟩
⟨datasection⟩ : : = [ ⟨text⟩ ]
⟨text⟩ : : = big ⟨string⟩; ⟨ftext⟩
           |⟨ftext⟩
⟨ftext⟩ : : = ⟨pointstohandler⟩ ⟨ftext⟩
            |⟨string⟩ ⟨ftext⟩
⟨pointstohandler⟩ : : = pointsto ⟨pth2⟩
⟨pth2⟩ : : = ⟨string⟩, ⟨pth2⟩
           |⟨string⟩
⟨pointedatbyhandler⟩ : : = pointedatby ⟨pbh2⟩
⟨pbh2⟩ : : = ⟨string⟩, ⟨pbh2⟩
           |⟨string⟩
⟨pageend⟩ : : = ⟨pointstohandler⟩ ⟨pageend⟩
              |⟨pointedatbyhandler⟩ ⟨pageend⟩
              |⟨null⟩
⟨pagefinish⟩ : : = endpage ⟨string⟩
```

APPENDIX 2

Synopsis of commands for the system

a [bsolute] page-number:
allows the user to select any page. The page is not displayed. If the user attempts to select a non-existent page, he is informed and the command aborted.

d [isplay] page-number:
displays the page requested.

m [ake] page-number:
makes a new page. The user is led through an interactive input session, which ensures a standard page appearance.

path:
Descends the hierarchy. If the user is currently on page '40.1', the command:

    # 2.4

selects page '40.1.2.4'

^ digit:
ascends the hierarchy by the number of levels given. If the user is on page '40.1.2.4', the command:

    ^ 3

selects page '40'.

p [ath name]:
prints the user's current position in the hierarchy.

l [ine-print] page-number:
prints the selected page on the line printer.

s [top]:
stops the session

w [here]:
displays an index of the pages below the current page.

h [elp]:
displays list of commands available.

APPENDIX 3

User session

It is very difficult to give the flavour of an interactive system on the printed page. Nevertheless, the following session is described in an attempt to convey the ease of scan and 'flick-through' the Vulcan system supports.

```
?: w          [display root index]

1 What's on in Town
2 Magazines – Current Issues
3 Television
4 Local Radio
5 BBC radio

?: a 1                    [select page one, but don't display]

?: w          [display page 1's Index]

1 London
2 Glasgow
3 Edinburgh

?: # 1        [hierarchically descend to page 1.1]

?: w

1 Cinemas
2 Theatre
3 Clubs and Pubs

?: # 1        [descend to page 1.1.1]

?: d          [display currently selected page]

Vulcan                                   Page 1.1.1

 XXXX  X X    X XXXXXX X    X   XX
X    X X XX   X X      XX  XX  X  X
X      X X X  X XXXXX  X XX X X    X
X      X X  X X X      X    X XXXXXX
X    X X X   XX X      X    X X    X
 XXXX  X X    X XXXXXX X    X X    X

In London
Key 'w' for index.
"Casablanca" at ABC 1.
See ABC 1 page for further information.
[This page is merely an advertisement]

?: w

1 ABC 5-Cinema Complex
2 Odeon 3-Cinema Complex

?: # 1              [Note that we have followed an extensive path, ]
                    [and the user is still dealing in single digits]

?: w

1 ABC–1
2 ABC–2
3 ABC–3
4 ABC–4
5 ABC–5

?: # 1

?: d

Vulcan                                        Page 1.1.1.1.1

  XX   XXXXX   XXXX         X
 X  X  X    X X    X       XX
X    X XXXXX  X             X
XXXXXX X    X X             X
X    X X    X X    X        X
X    X XXXXX   XXXX        XXX

Casablanca now showing.
Starring Humphrey Bogart and Ingrid Bergman.
Continuous Performances, Full supporting Program.
Doors open 1.00a.m. Program Starts 1.20a.m.

?: w

nowhere                   [The user has reached a terminal page]

?: ^ 1                 [He rises one level of the hierarchy]

?: # 2                 [He selects a different branch]

?: d

Vulcan                              Page 1.1.1.1.2

  XX   XXXXX   XXXX        XXXX
 X  X  X    X X    X      X    X
X    X XXXXX  X               X
XXXXXX X    X X             XX
X    X X    X X    X       X
X    X XXXXX   XXXX       XXXXXX

White Heat
Starring James Cagney
Doors open 7.00p.m. Program starts 7.30p.m.

?: p  [The user wants to know exactly where he is]

1.1.1.1.2

?: ^ 7                 [The user rises 7 levels. As he is only down
                       [5, he is taken as high as he can go          ]

?: w                   [Root Index]

1 What's going on in town
2 Magazines – Current Issues
3 Television
4 Local Radio
5 BBC Radio
```