VISUALIZATION OF PROGRAMS IN TEXTBOOKS

Pieterse V and Bishop JM
Department of Computer Science, University of Pretoria
May 1996

The analysis for a generic design of a hypothetical system that is called VIZAUTHOR is discussed in this paper. It is an open integrated visual programming environment that possesses all the necessary tools to develop and maintain visual books. It is highly adaptable and therefore very versatile. It can also be used for literate programming or for the development and maintenance of hypermedia databases.

The term visual book refers to an electronic textbook which can be used to teach the reader programming design concepts as well as programming in a specific programming language.

Various specialised topics, each of which may contribute in a special way to the design of the proposed environment with its unique properties, were identified. The best features of several environments that originated from divers subjects within computer science, must be consolidated in VIZAUTHOR. The fields of study involved are *Literate Programming, Visualization of programs, Visual programming, Integrated Programming Environments and Hypermedia.*

1. Introduction

1.1 Computer based training

Computer Based Training (CBT) can have a meaningful positive influence on the efficiency in which the learners retain knowledge after learning. Reinhardt (1995) refer to various experiments that confirm this assertion.

Contributions that are made towards more effective learning and teaching with the aid of computers are reported regularly at the bi-annual conference on multimedia an hypermedia that is held at the University of Pretoria. (Britz 1996; Ritchie 1994; Cronjé 1996; Hall 1994)

1.2 Visualization

It is well known that visualization plays an important role in education and training. Authors such as (Bork, 1975), (Mayer, 1981), (Brown & Cunningham, 1990), (Cunningham et al, 1990) and (Quinn, 1991) have shown by experiment that the use of visualizing has a meaningful influence on actual learning, where the capability of the learner to apply the obtained knowledge to new problems is used to measure learning. The use of diagrams and animations to visualize the different aspects of computer programs will attribute to the value of visual books as learning aids.

1.3 Availability

The possibility of using computers to enhance learning has been extended by recent developments in computer technology. Both software and hardware needed for successful use of computers in teaching is more readily available than few years ago. A study conducted by Bender (1994) showed that the use of computers in South-African schools has increased dramatically since 1990. The quality and quantity of available educational software is accelerating.

1.4 Motivation

In the presence of evidence that CBT can enrich learning, it is considered worth the trouble to investigate in the use of the newest technology to improve the quality of teaching. The discussion of existing environments shows that the creation of an environment that is favourable for learning programming and programming design is a realistic goal. The intended learning environment will in this document, be referred to as a **visual book**.

2 Research

While looking for an environment that has enough tools to create a visual book, several topics of computer science have been considered.

It was found that computer scientists are inclined to create new environments to meet specific needs rather than to use existing environments. The result is a vast number of custom made environments which are not useable outside the intended need. Many of these do have excellent tools suitable to create a visual book. Unfortunately most of them are inadequate in some or other way due to the fact that they were created with some other goal than a visual book in mind.

3 Existing Environments

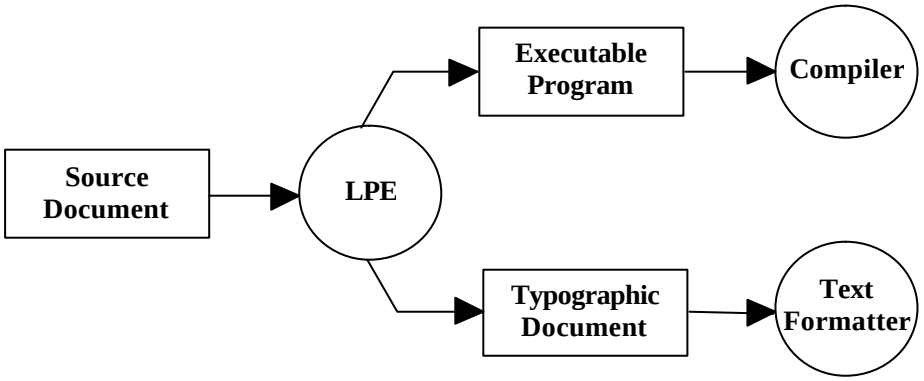
3.1 Literate Programming Environments (LPE's)

3.1.1 Introduction

Knuth (1984) coined the term "*Literate Programming*" to describe a programming paradigm. According to this paradigm the programs should be written in a style which makes the reading of a program an enjoyable experience. The structure of the program must at all costs enhance the human understanding of the program. Literate programming is thus a way to design and document programs primarily to explain to a human what the computer is supposed to do. This is radically different from the traditional viewpoint of a program as a sequence of instructions to a computer to solve a problem.

3.1.2 Structure

The following diagram illustrates the structure of a generic LPE. There are LPE's that deviate from this structure, but the majority is based upon it.



A source document of the literate program is created in the LPE. The LPE uses two different processes to generate an executable program document and a typographic document from the source document. These documents is respectively executed by a compiler, and printed with a text formatting tool.

3.1.3 Good properties

A visual book can be seen as a collection of literate programs. The following are properties of literate programs that a visual book should have:

- **Pretty Printing**

Pretty printing refers to the automatic use of fonts, text colours and layout to enhance the readability and understanding of code. Use of pretty printing is a matter of course in modern programming environments. In his article on Paradox 7, Riccardi (1996) mentions that the leading editors uses pretty printing to visualize the meaning of sentences.

As in LPE's and modern programming environments, VIZAUTHOR will automatically apply pretty printing

to all code in a visual book. This will ensure that the code in a visual book is formatted uniformly and readable.

- **Verisimilitudity**

This property identified by Thimbleby (quoted by van Wyk, 1990) is concerned a precondition for true literate programming. This requires that both the executable program and the typographic document must be automatically generated from exactly the same source document of the literate program. In the LPE's studied, this is accomplished with first level data integration (§3.4.2).

With VIZAUTHOR it will be possible to trace through code, view the same code in WYSIWYG-mode and to print it with a state of the art desktop publisher. All this can be done by using third level data- and control integration.

- **Psychological arrangement**

The structure and ordination of a literate program is dictated by psychological terms and should not be influenced by syntactic concerns (Brown and Childs, 1990). This property distinguish literate programs from highly documented programs. Freedom of arrangement as implemented in LPE's can be used to design a visual book with the most logical structure from an educational point of view.

3.1.4 Constraints

The following are properties of LPE's which hinder the creation of visual books by using a LPE alone:

- **Code can not be traced interactively**

Most LPE's uses a preprocessor to extract the code from the source document and produce it in a executable format to a compiler to execute it. Not one of the LPE's studied, have tools to trace or step through the generated code, let alone tools to trace the code in the source document.

Generally it is hard to determine the origin of code that caused an error when the generated program is executed in a LPE. Interactive stepping and tracing through code has great value for debugging and is vital in a visual book to visualize the execution of the code.

- **Not adapted for documents that may contain multiple programs and partial programs.**

The LPE's studied generates one complete executable program at a time, using the hole source document. This is an unacceptable constraint on a visual book. It must be possible to have multiple programs and partial programs in a visual book.

- **Not enough visualization**

In most LPE's pretty printing is the only form of visualisation found. Some LPE's like LIPED (Bishop and Gregson, 1992) and WHS (Brown and Czejdo, 1990) visualizes the program structure in a diagram mainly to ease up navigation.

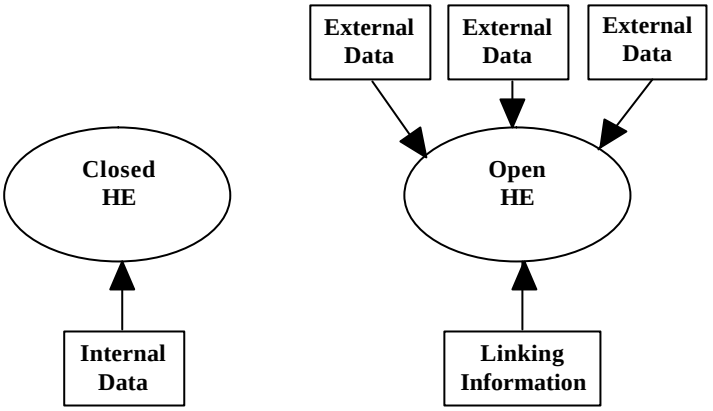
3.2 Hypermedia Environments (HE's)

3.2.1. Introduction

Hypermedia can be defined as multimedia containing hyperlinks. Multimedia is the use of multiple media in one document. A multimedia document can thus contain text, graphics, sound, music, animation, video, virtual reality, etc. Hyperlinks enable the reader to explore related data with the click of a mouse. Akscyn (1994) calls hypermedia documents "Active documents".

3.2.2 Structure

The following diagram depicts the generic structures of respectively a closed HE, and an open HE.



In a closed HE all data is stored in an internal format and can only be manipulated using the HE's editors.

Open HE's uses external data sources which can not be edited within the HE. Linking information is either standardised (e.g. HTML) or handled internally.

3.2.3 Good properties

A visual book can be seen as a hypermedia document and must have the following hypermedia properties:

- **Use of multimedia**

More media means more methods can be used in explanations in a visual book. Woolf and Hall (1995) say that it can create a stimulating and motivating learning environment.

- **Hyperlinks and guided tours**

Hyperlinks induces a non-sequential order. According to Romiszowski (1990) their use can promote learning if it is correctly used. A guided tour is like a tutorial which can reveal the necessary precognition as well as the essence for the understanding of a topic in a psychological correct order.

- **Intelligent interaction**

The fact that artificial intelligence can be built into a hypermedia document, makes it an appealing medium for education and training. (Louw, 1994), (Woolf and Hall, 1995).

3.2.4 Constraints

The following constraints of HE's inhibits the creation of a visual book with the aid of a HE alone:

- **Importability**

The most cited problem with HE's is it's complex and private data structure which makes it importable to other HE's or platforms. This undesirable situation is commonly referred to as the "docu island"-problem and was identified by van Dam (1988) in 1987.

Recently open hypermedia environments such as Microcosm (Rake and Davis, 1994) have been developed, but there is still many problems in connection with the editing of external documents and updating of links because external documents can be edited by external editors without the awareness of the HE.

- **Code in a HE is not executable**

Normally code cannot be executable in a HE due to the complex internal data structure required by the HE.

- **Intricate visualization of code**

Although HE's are build to contain animation and graphics, no general HE that contain tools to automatically visualize any aspect of a computer program could be found.

3.3 Visual Programming Environments (VPE's)

3.3.1 Introduction

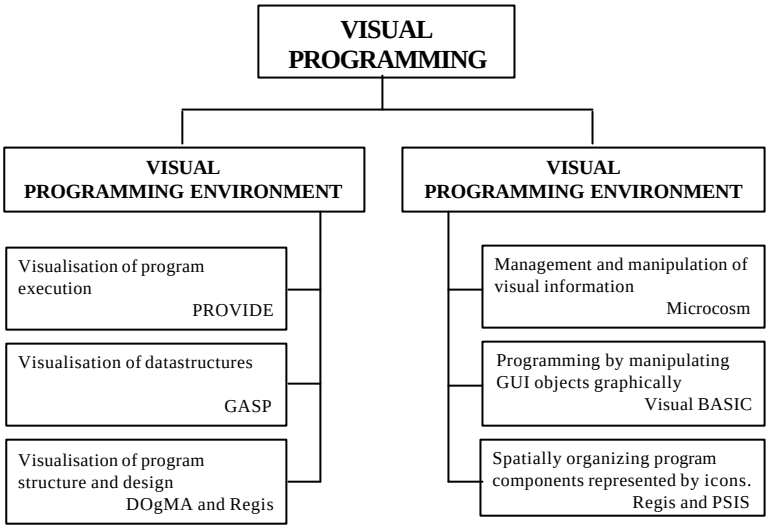
Visual programming is the use of visual expressions such as icons or diagrams, or graphical manipulation during programming. According to Burnett and McIntyre (1995) visual programming can be done in a visual environment or by using a visual language. This distinction is illustrated in the accompanying diagram.

Visual Programming Environments

A visual programming environment contains tools to visualize some aspect of a program irrespective if the language in which the program is written, is textual or visual.

Visual Programming Languages

A visual language uses graphical manipulation to create programs. If the syntax of a language contains any form of visual expression, it is called a visual language.



The following systems is mentioned as examples in the diagram in the given categories:

- **PROVIDE (Moher, 1988)**

  A process visualization and debugging environment that uses computer graphics to depict process states which can directly be manipulated by the user.

- **DOgMA (Sametinger and Schiffer, 1995)**

  A programming environment that supports visualization of dynamic aspects, such as animating objects and their interactions.

- **Regis (Magee et al, 1994)**

  A programming environment in which the description of program structure is separated from the programming of computational components.

- **Microcosm (Hall, 1996)**

  An application that uses generic links to process queries in a database that may contain dynamic multimedia objects.

- **Visual BASIC (Linthicum, 1996)**

  A successful commercial VPE by Microsoft that provides a suite of visual programming tools including a window painter, an application builder and an interactive graphical debugger.

- **PSIS (Chang S-K et al, 1995)**

  Pittsburgh-**S**alerno **I**conic **S**ystem is a system in which the programmer design visual sentences by manipulating icons on the screen.

3.3.2 Good properties

The following advantageous properties of VPE's were identified

- **Code can be traced interactively**

All the studied VPE's can visualize the execution of their programs by means of an interactive debugger. This tool enables the user to execute a program step-by-step while monitoring and manipulating the values of variables in the program.

Stepping and tracing can be utilised in a visual book to enforce the reader's understanding of a working program.

- **Visualization**

Brook (1987) describes the capability of the human mind to interpret visual information as mans most powerful conceptual tool. The use of visualization in a visual book will therefore enable the reader to make better use of his own conceptual ability. The techniques used for visualization in VPE's is mostly excellent.

- **Class Browsers**

Sametinger and Schiffer (1995) remarks that file browsers that were adequate before the Object Oriented (OO) era are now obsolete. OO programming is unthinkable without a class browser. All die PE's for OO programming that was studied, uses multi-paned windows for class browsers.

- **Query By Image Content (QBIC)**

The QBIC-technology analyses properties such as colour, texture, form, position, etc. of an image and determines numeric values of measurement for all the properties. The results can be used to decide if a given image answers to the criteria of the query

- **Dynamic updating of associated views**

Most VPE's update associated diagrams and code dynamically. This means that if one of several associated views are changed all the other views are updated according to the change to keep views consistent with each other.

3.3.3 Constraints

The following constraints of VPE's was identified:

- **Only one-way automatic updating of associated views.**

Unfortunately dynamic updating of associated views is usually only done in one direction. For example if a diagram is dynamically updated if the associated code is changed, it will normally not be possible to edit the diagram. If the diagram can be edited, the code is hardly ever updated accordingly.

- **Rigid program structure**

Most VPE's does not cater for the programmers own documentation other than remarks within the code or captions to generated diagrams. Probably this is so because designers of VPE's may feel that the documentation that is automatically created is more than adequate.

- **Limited aspects visualized.**

Most of the VPE's that was studied, only visualize singular aspects of programs. In spite of this fact, a large amount of aspects of programs to visualize and ways to visualize them was found within the set of VPE's that were studied.

3.4 Integrated Environments (IE's)

3.4.1 Introduction

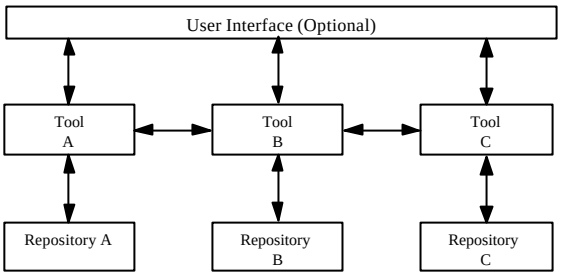
An integrated environment is an environment which integrates tools and applications needed in different stages of a project for ease of use.

3.4.2 Levels of integration

Sharon and Bell (1995) defined the following levels of integration of tools in an environment.

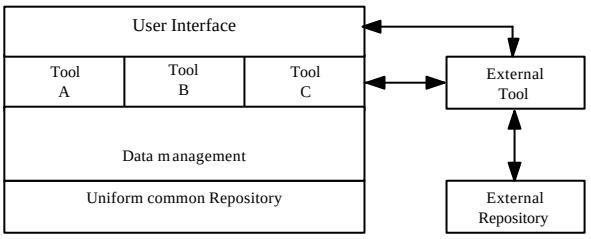
- **First level integration**

The tools operate separately, each on it's own internal data. Data is not shared but merely transferred from one tool to the other. They are only linked through common import and export formats. Transferring of data from one tool to the other is done by processes that are invoked manually by the user.



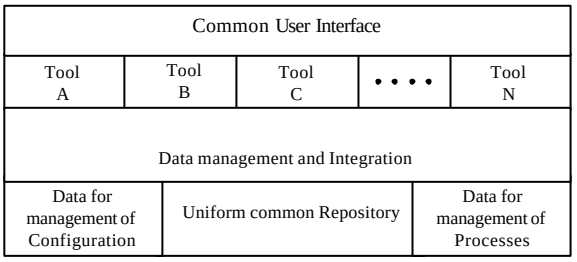
- **Second level**

Tools are fully integrated and all data is stored in an uniform format in a common repository. The use of the shared data by different tools is transparent to the user. A tool from other vendors can only be integrated if it uses the same metamodel. Some vendors publish the metamodels for data storage to encourage the development of third-party tools that can be integrated into their environments.



- **Third level**

Third party tools are bound in a framework with a common user interface. All the tools use the common repository which is managed by the framework. Transferring of data from one tool to the other is done automatically by processes that act transparently to the user.



3.4.3 Good properties

- **Common user interface**

The ease of use is enhanced by uniform ways to activate applications, use tools and manipulate data.

- **Data Integration**

Data integration deals with representation, conversion, and exchange of data in a common metamodel. Bell and Sharon (1995) say that integration has become practical due to recent standards and popularity of integration frameworks.

- **Control Integration**

Control integration deals with the updating of associated data from different tools. In OO programming this can be done by passing of messages. Taschek (1995) explains how IBM's *"Common Object Request Broker"* CORBA technology and the *"Object Linking and Embedding"* OLE technology of Microsoft both boil down to client/server technology. The use of control integration between external applications to create, edit and manipulate the data in a visual book will eliminate the need to create editors and technology-specific tools to be used in VIZAUTHOR.

3.4.4 Constraints

- **Limited dependency links**

In the IE's studied it was found that linking between different tools was mostly established by embedding the data of one tool within the data of the other. In VIZAUTHOR there is a need for two-way dependency links to invoke automatic two-way updating of associated objects in a visual book.

- **Limitation to certain tools**

With second level integration the tools that can be used is limited to the tools of the supplying vendor. Third level integration in theory give a wider range of applications and tools to choose from, but in practice the integration of a new tool remains time consuming and tricky.

- **Duplicating of data**

With first level integration shared data is often stored in the native form of both applications which shares the data, as well as in a common portable format. Unilateral changing of data by a application can easily lead to inconsistent data.

4. Design

VIZAUTHOR must be an open, fully integrated, adaptable, literate programming environment which uses hypermedia and mainly third party tools to visualise programs in order to teach programming design concepts as well as programming in a specific programming language.

4.1 Open

The ease of transferring data between a system and other systems determine the openness of a system. If the system's internal data can be transferred to other systems, and if data from other applications can be easily used, the system is open.

4.2 Integrated

It is obvious VIZAUTHOR must be based upon an established third level integration standard that is able to provide a vendor-neutral programming interface for integrating all the different tools needed.

4.3 Adaptable

VIZAUTHOR must be independent of any formatter, programming language or specific tool. It can be seen as an ideal toolbox which can be used to easily integrate the necessary tools in such a manner that any tool can be added, withdrawn or replaced at any stage.

4.4 Literate Programming Environment

The data structure of documents created in VIZAUTHOR must alow psychological ordering. Pretty printing must be applied. The table of contents, index and cross references must be created automatically. A method must be found not only to generate the literate version and the executable version of a program from the exact same source, but also to be able to interactively debug the literate version of the program.

4.5 Hypermedia

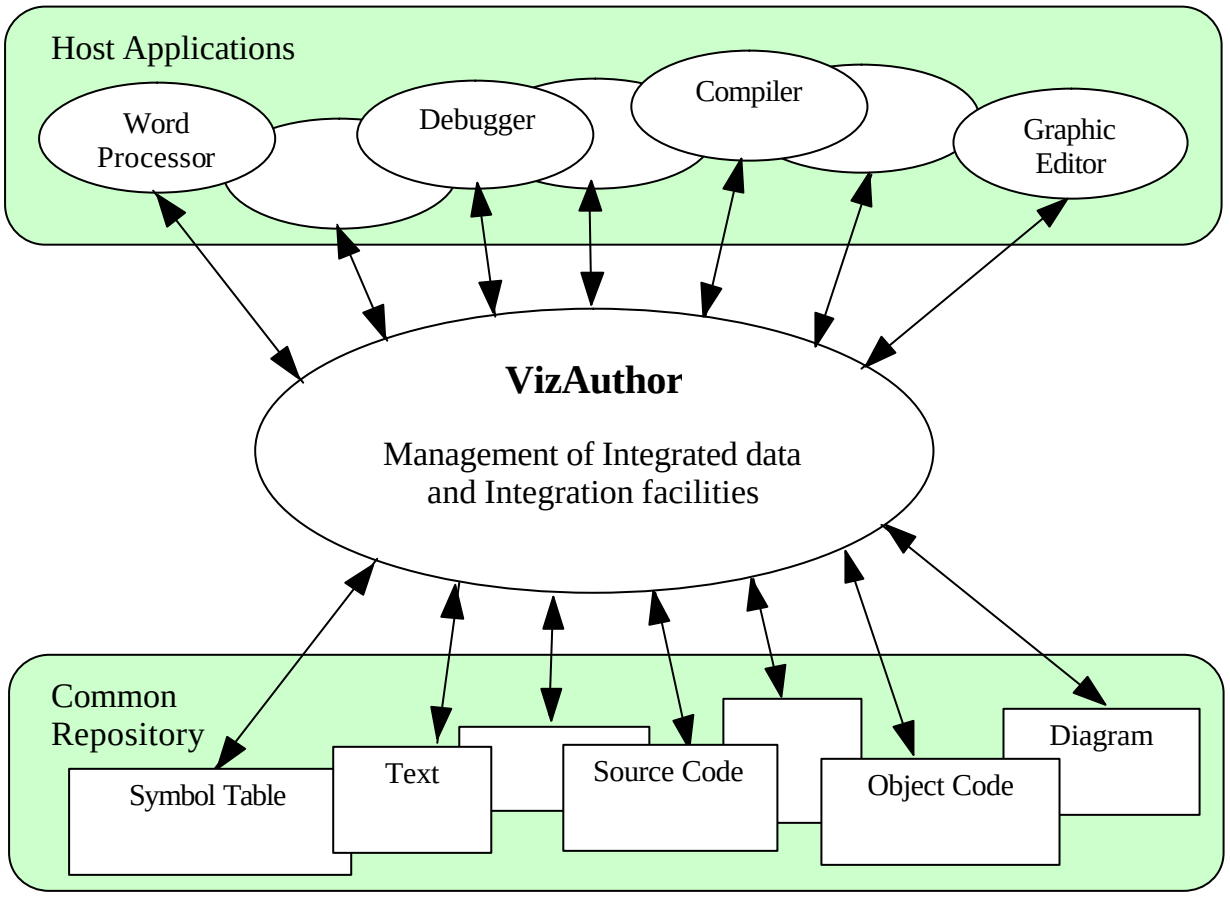
VIZAUTHOR must provide interfaces to easily integrate all known media. The use of hyperlinks and guided tours is essential.

4.6 Third party tools

It will be counterproductive to try develop tools for VIZAUTHOR which has already be done on a very high standard by others. VIZAUTHOR must provide means to utilise the vast source of existing tools and tools that will be developed in the future optimally.

4.7 Visualising

In his famous *"No silver bullet"*-paper, Brook (1987) said that software is invisible and cannot be visualised because of it's complexity. Although it is indeed true that it will be impossible to visualise a invisible abstract item such as a computer program, there is many ways to effectively visualise some aspects of a program. VIZAUTHOR must integrate tools that can produce these automatically.



The design of VizAutor

5 Summary

The following table shows to what degree the different types of environments supports the mentioned useful properties that VIZAUTHOR should have:

	TPE	LPE	HE	VPE	IE
Uniform User Interface	●	○	◉	◉	●
Adaptable	○	◉	○	○	◉
Multiple programs in one document	●	○	●	○	●
Use of Multimedia	○	○	●	◉	●
Use of third party tools	○	○	◉	◉	●
Dependency	○	◉	○	◉	◉
Integration of data	○	◉	○	◉	●
Switching between views	○	◉	◉	●	●
Hyperlinks and guided tours	○	○	●	○	○
Navigation in program structure	○	○	◉	◉	◉
Intelligent interaction	○	○	○	○	○
Freedom to order Psychologically	○	●	●	○	○
Compilation of psychologically ordered code	○	●	○	○	○
Visualization of program execution	●	○	○	●	◉
Visualization of data and data structures	○	○	◉	◉	◉
Visualization of program structure	○	◉	◉	●	◉
Visual methods to program	○	○	○	◉	◉
Manipulation and management of Visual information	○	○	●	●	◉
Generation of cross references and indexes	○	●	◉	○	○
Automatic generation of documentation	○	◉	○	◉	◉
Automatic pretty printing	○	●	○	○	○

Legend

TPE	Text oriented programming environments
LPE	Literate programming environments
HE	Hypermedia editors
VPE	Visual programming environments
IE	Integrated environment

○ Never
○ Sometimes
◉ Often
◉ Mostly
● Always

6. Conclusion

While studying the different environments mentioned in this paper, it was significant that, although there is much in common between the different types of environments, they have not borrowed from each other's experiences. Every type of environment was custom made with only its own goal in mind. Therefore most environments can not be used outside the domain in which they were created.

With this observation in mind, VIZAUTHOR must be created in such a manner that it will be able to contain not only all the tools that are concerned vital for the creation of visual books, or the tools that will not constrain future ideas for visual books, but also the superset of tools identified within the studied environments to create a universal adaptable environment that can be used across the various specialised topics.

References

1. Akscyn RM. *Re-engineering the field: Hypertext in the 21st century*. HYPERMEDIA '94 Proceedings of the 2nd Southern African Conference on Multimedia an Hypermedia. p. 1-10 (23 - 25 March 1994)

2. Bell R, Sharon D. *Tools to Engineer New Technologies into Applications.* IEEE Software **12** (2) : 11-16 (March 1995)

3. Bender JM. *Factors that determine successful computer implementation in schools*. M Ed Thesis, University of Pretoria (1994)

4. Bishop JM; Gregson KM. *Literate Programming and the LIPED Environment*. Structured Programming. **13** (1) pp 23-34. (1992)

5. Bork A. *Learning throug graphics*. Ten-year forcast for computers and communications : implications for education. HumRRo Conf (Sept 1975)

6. Britz B. *A case study of the use of multimedia and hypermedia in the corporate training environment*. HYPERMEDIA '96 Proceedings of the 3rd Southern African Conference on Multimedia an Hypermedia. University of Pretoria. (May 1996)

7. Brook FP Jr. *No silver Bullet : Essence and Accidents of Software Engineering*. Computer. **20** (4) : 10-19. (April 1987)

8. Brown JR; Cunningham S. *Visualization in Higher Education*. Academic Computing (March 1990)

9. Brown M; Childs B. *An Interactive Environment for Literate Programming*. Structured Programming, **11** (1) : 11-25. Springer-Verlag. (1990)

10. Brown M; Czejdo B. *A Hypertext for Literate Programming*. Advances in Computing and Information ICCI '90. International Conference Proceedings. pp 250-259. (23-26 May 1990)

11. Burnett MM; McIntyre DW. *Visual Programming*. Computer **28** (3) : 14-16. (March 1995)

12. Chang S-K, Costagliola G, Pacini G, Tucci M, Tortora G, Yu B, Yu J-S. *Visual-Language System for User Interfaces.* IEEE Software **12** (2) : 33-44 (March 1995)

13. Cronjé J. *Threading a web for training*. HYPERMEDIA '96 Proceedings of the 3rd Southern African Conference on Multimedia an Hypermedia. University of Pretoria. (May 1996)

14. Cunningham S; Brown JR; McGrath M. *Visualization in Science and Engineering Education*. in **IEEE Tutorial : Visualization in Scientific Computing.** Edited by Nielson GM, Shriver B. IEEE Press (1990)

15. Louw WJA. *Hypertext and its use in education*. HYPERMEDIA '94 Proceedings of the 2nd Southern African Conference on Multimedia an Hypermedia. University of Pretoria. p. 117-124 (March 1994)

16. Hall W. *Ending the tyranny of the button: Extending the role of hypermedia in education and training*. HYPERMEDIA '94 Proceedings of the 2nd Southern African Conference on Multimedia an Hypermedia. University of Pretoria. p. 11-18 (March 1994)

17. Hall W. *Zen and the Art of Linking*. in **HYPERMEDIA '96 Proceedings of the 3rd Southern African Conference on Multimedia and Hypermedia** (23 - 24 May 1996)

18. Knuth DE. *Literate Programming*. The Computer Journal, **27** (2) : 97-111. (1984)

19. Linthicum DS. *Visual Basic 4.0: ready for the enterprise*. DBMS **9** (1) : 44-50. (Jan 1996)

20. Magee J, Dulay N, Kramer J. *Regis: a constructive development environment for distributed programs.* Distributed System Engineering **1** : 304-312. The British Computer Society, The Institution of Electrical Engineers and IOP Publishing Ltd. (1994)

21. Mayer RE. *The Psychology of How Novices Learn Computer Programming*. Computer Surveys, **13** (1) : 121-141 (March 1981)

22. Moher TG. *PROVIDE : A Process Visualization and Debugging Environment* IEEE Transactions on Software Engineering. **14** (6) : 849 - 857 (June 1988)

23. Quinn G. *Encoding and maintenance of information in visual working memory*. in **Mental Images in Human Cognition.** Edited by Logie RH, Denis M. Elsevier Science Publishers B.V. (North-Holland) (1991)

24. Rake S, Davis H. **Microcosm: Release 2.2 Documentation.** Department of Electronics and Computer Science, University of Southampton. UK (1994)

25. Reinhardt A. *New Ways to Learn*. BYTE **20** (3) : 50 - 72 (March 1995)

26. Riccardi S. *Paradox 7: Powerful 32-bit Database Build For Developers*. PC Magazine South Africa. **4** (2) : 23,28 (March 1996)

27. Ritchie I. *Hypermedia in education - will the promise ever be fulfilled ?* HYPERMEDIA '94 Proceedings of the 2nd Southern African Conference on Multimedia an Hypermedia. University of Pretoria. p. 19 - 25 (March 1994)

28. Romiszowski A. *The Hypermedia/Hypermedia Solution - But What Exactly is the Problem ?*. In **Designing Hypermedia for Learning.** edited by Jonassen DH and Mandl H. Springer-Verlag. (Berlin) p. 321 - 373 (1990)

29. Sametinger J; Schiffer S. *Design and Implementation Aspects of an Experimental C++ Programming Environment*. Software - Practice and Experience. **25** (2) : 111 - 128 (Feb 1995)

30. Sharon D, Bell R. *Tools That Bind: Creating Integrated Environments*. IEEE Software **12** (2) : 76-85 (March 1995)

31. Taschek J. *IBM, Microsoft share client/server vision* in PCWeek executive edition South Africa, **1** (2) : 23-24 (9-22 Oct 1995)

32. Van Dam A. *Hypertext '87 Keynote Address*. Communications of the ACM, **31** (7) : 887 - 895. (July 1988)

33. Van Wyk CJ. *Literate Programming : An Assessment* In Literate Programming. Communications of the ACM, **33** (3) : 361-365. (March 1990)

34. Woolf BP, Hall W. *Multimedia Pedagogues : Interactive Systems for Teaching and Learning*. Computer **28** (5) : 74 -80 (May 1995)