The International JOURNAL of LEARNING

Design of a Web-based Visual Learning System (WVLS) via Plan Across Disciplines

Alireza Ebrahimi

VOLUME 12, NUMBER 10

COMMON GROUND

www.Learning-Journal.com

INTERNATIONAL JOURNAL OF LEARNING
http://www.Learning-Journal.com

First published in 2005/2006 in Melbourne, Australia by Common Ground Publishing Pty Ltd
www.CommonGroundPublishing.com.

ISSN: 1447-9494 (print), 1447-9540 (online)
Publisher Site: http://www.Learning-Journal.com

The INTERNATIONAL JOURNAL OF LEARNING is a peer refereed journal. Full papers submitted for publication are refereed by Associate Editors through anonymous referee processes.

Typeset in Common Ground Markup Language using CGCreator multichannel typesetting system
http://www.CommonGroundSoftware.com.

Design of a Web-based Visual Learning System (WVLS) via Plan Across Disciplines

Alireza Ebrahimi, State University of New York / Old Westbury, United States of America

Abstract: Experiment the design and re-design of a Web-based teaching and learning system by visualization of plans and language constructs. WVLS is a visual teaching learning that can be designed easily by educators and is based on visualization, abstraction, plan integration and plan relationship to understand a concept or solve a problem not necessarily mathematical. The center of WVLS is a plan representing the main concept or a problem which after exploration of plan and interaction eventually will lead to its solution. A problem is divided into sub problems, as much as needed making a problem and its solution understandable and manageable. At each stage two plans are integrated to form a new plan with four integration methods known as append, embed, interleave, and branch. WVLS consists of three separate phases known as observation, integration, and creation. WVLS effectiveness has been successfully tested on learning subject such as programming as well as math, science and other fields.

Keywords: Web-based Design, Teaching and Learning System, Plan and Integration, Concepts and Problem Solving, Standard Language Constructs, Technology in Learning

Introduction

WEB BASED LEARNING systems are becoming common ground for teaching and learning [Arnow, 99; Truong, 05]. A sound design model has the potential to significantly increase the effectiveness of these systems. Teaching and learning can be enhanced when a problem and its solution are visually decomposed into plans and its sub-plans and subsequently integrated. A plan is an abstraction that visually provides a solution to a problem or to a sub-problem in the sense, it represents a problem from its macro level down to its micro level. Any important concept can be described as a plan and it is up to the educator and level of learner to decide on a particular plan. A plan can be visually represented by a dot, a geometric shape or an image. The process of visualizing plan decomposition and then plan integration for a problem is demonstrated in WVLS – Web Visual Learning System.

WVLS divides the process of learning and its enforcement into three selectable phases known as plan observation, plan integration and plan creation. WVLS initially provides a library of sample problems (plans) working with all three phases. A learner can observe the process of solving a problem, become involved in a partial solution, or solve the problem entirely from beginning to end. A mixture of learning strategies and techniques are incorporated in WVLS to satisfy a wide range of learners.

WVLS will identify and report the cause of problems to the learner. A systematic approach to analyse a solution based on plan relationships indicates whether a plan is missing, misplaced, malformed or has a misconception. The effectiveness of a similar visual plan environment, known as VPCL, as a learning and instructional tool has been proven by the result of empirical studies on novice programmers.

Web Tools for a Learning Design

Today, there are a number of web tools that are available to design a learning system, like any other web pages. With the current technology, learning a web tool should be as easy as learning word processing. A common available web tool could be Microsoft Office FrontPage or Microsoft Word, both of which provide functionality necessary to save the document as a web page. More sophisticated systems, such as Dreamweaver and Flash from Macromedia Company, can be employed when a higher level of visual stimulation is required.

Web authoring tools such as, FrontPage or Word are defined as such because they automatically generate HTML (Hyper Text Mark up Language) tags, the declarative language that describes every page visible on the World Wide Web [Niederst, 2001]. To view a webpage's HTML code, one can simply right click their mouse. It is a good gradual learning to view the source code of a webpage. By viewing the source code of one's favorite view webpage, it is possible to figure out some of the styles and ap-

proaches in the design of a learning system. While it is not necessary to learn HTML tags to build a learning system, learning (memorizing) the 10 important commands of HTML makes up about 90 percent of all web page design [Ebrahimi, 2002]. There are even beginners who prefer HTML over other software since it is easy to manipulate and it is available on any system. All that is necessary to accomplish this is a text editor such as Notepad and an internet browser such as Explorer, both of which can be readily obtained at no cost.

By adding a little program, which is called Script, to HTML, it is possible to make the learning system powerful and innovative. JavaScript is the dominant web scripting language, with VBScript trailing a distant second. An individual may not know client-side language to become proficient in, but a little knowledge of copy and paste of common built-in script functions to the learning system can make a difference. Many novice designers become frustrated and overwhelmed with the current web tools and surrounding technology. The same group has also enjoyed the learning and experience as a result. While some stayed with their own design, others asked professional designers to design the new learning system – synergizing the benefits of an experienced educator with that of a seasoned software developer, inevitably resulting in a more robust learning platform.

An essential lesson to learn while designing a web learning system is to separate what is important and what is less important. Designers should focus on the major issues rather then getting involved in the jargon of instructions and details. Figure 1 illustrates the HTML and JavaScript code used to build a sample page from WVLS.

```
<html>
<script language = "JavaScript">
function cin(){
   alert("cin>>5;"); }
</script>
<body><center>
<h1>VPCL PLAN LIBRARY</h1><hr width=75%>
<h1>Find Average Problem</h1><h1>Phase 1: Plan Observation </h1>
<h1>Program Execution</h1>
<table border="0" width="42%">
<tr><td >num</td><td >sum</td><td>count</td></tr>
<tr><td ><input type="text" size="4"></td><td><input type="text" size="4"></td>
<td align="center"><input type="text" size="4" value="0"></td></tr>
</table>
<form action="findavg/obsv/o8.htm">
<input type="submit" value="Loop" onclick="cin()">
</form>
</center></body></html>
```

Figure 1 - HTML and JavaScript Example

Problem, Plan, Visualization, and Web

The common trend in solving a problem is to break a problem into its sub-problems and solve each sub-problem separately. Breaking down the problem makes it easier to understand and easier to solve. Therefore, a plan to solve a problem is set aside and sub-plans are made for sub-problems. Once the plan is broken down into sub-plans, then each sub-plan can be addressed separately. A plan, such as building a house, itself may consist of other plans, such as plans for doors and windows, which also may be composed of sub-plans. The sub-plans are then integrated one by one to form the final plan, which is also the final solution. The following steps are necessary in building a learning system: (1) identify your plan, (2) explain the plan using a easily understood semantic language, (3) propose a solution plan, (4) implement the solution plan, and (5) establish a channel for feedback, such that solutions can be revised as necessary. The idea of plan in artificial intelligence and learning theory can be traced back to Roger Schank (1975) who introduced the concept of script which is a general plan for a series of actions.

Visualization plays an important role in illustrating a plan, sub-plans and their relationships (i.e. plan hierarchy and plan structure). Depending on the problem and the level of the learner, a plan can be further divided into sub-plans until there is no need for clarification and the problem is understood. A plan is an abstraction that represents a solution to a problem; it deals with the entire solution or the smallest component that is crucial to solving a sub-problem. The plan abstraction applies information hiding so that the learner is not overwhelmed with

information that is not necessary at the moment. In order to systematically solve a problem visually using plan decomposition and plan integration, an environment known as WVLS - Visual Plan Environment has been implemented. WVLS provides an incremental way of learning and problem solving, rather than requiring the problem to be solved all at once. WVLS assures the learner's capability and assesses the learner's problem at an early stage and grows as the learner grows. To understand and learn how a problem is solved, WVLS is divided into three phases of learning: plan observation (rehearsal), plan integration (composition), and plan creation (new plan).

An example of a graphical and visual environment built to help learners understand difficult materials is Alice for learning 3D graphic programming [Conway et al, 00]. Another example of an interactive web-based system is Environment for Learning to Program (ELP), which helps teaching programming to the novice by not having the need for Program Development Environment and installation of programming language. Using ELP, students use template exercises through web to build their programming and problem solving skills [Truong, 2005].

Phase 1 of Learning: Plan Observation

Learning can be enhanced if the learner can observe how a problem is solved step by step and be able to repeat the observation over and over until it is understood. Plan observation enforces the retention of the plan and enables the learner to relate it to other similar situations. The WVLS plan observation phase is an automatic process that illustrates the steps involved in a task, starting with the initial specifications of the problem to its final solution, from generalization to specification. In the plan observation phase, the learner walks through the entire process of problem solving including plan decomposition and plan integration and can repeat the process as many times as necessary. An example of this phase would be for a child to watch the steps in building a house or bridge using blocks. Similarly, a novice programmer can observe how a group of numbers is sorted.

Phase 2 of Learning: Plan Integration

A solution to a problem consists of several sub-plans that need to be integrated correctly. After decomposing each sub-plan, it is important for the learner to understand the relationship between the sub-plans, such as how these sub-plans are put together, their spatial relationships and their order. There are systematic ways to integrate two plans. For example, in WVLS there are four ways to integrate plans: an appended plan (plans are one after the other sequentially e.g. steps); an interleaved plan (go back and forth between plans, e.g. relationship between day and night); a branched plan (a plan can be diverted to other plans based on some condition e.g. a plan will lead to one plan or another); or an embedded plan (a plan can be entirely embedded within another plan, e.g. Russian dolls). Plan integration is a good test to ensure and examine the understanding of the solution of the problem at an abstract level, rather than its detailed steps.

For example, a plan can be designed to solve a mathematical expression, applying precedence of operators. To solve the expression: 4 * 7 + 20 / 5, a sum plan can be created with two sub-plans, multiply and divide. These plans must be integrated a certain way so that the final solution yields the correct answer. One way to solve this problem is to have the multiply and divide plans appended to each other and embedded in the sum plan.

Plan Integration Modes

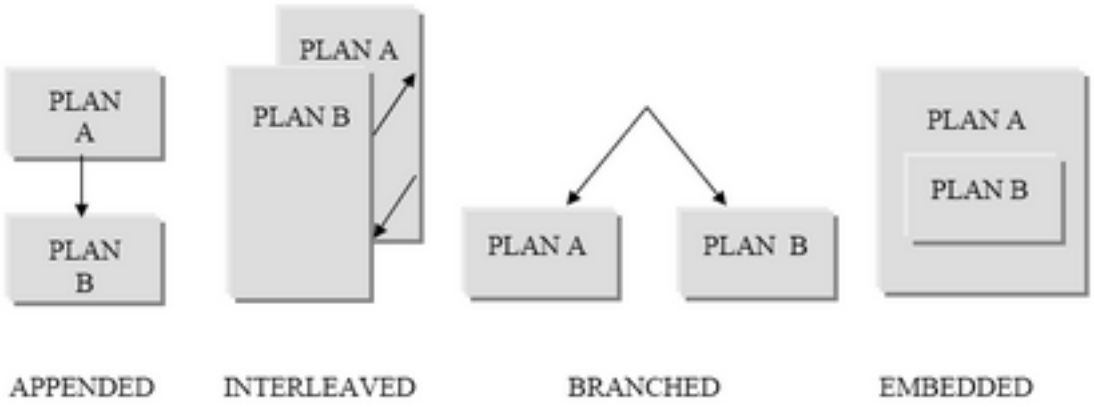

Figure 2: Plan Integration Modes

Find Average Plan Example

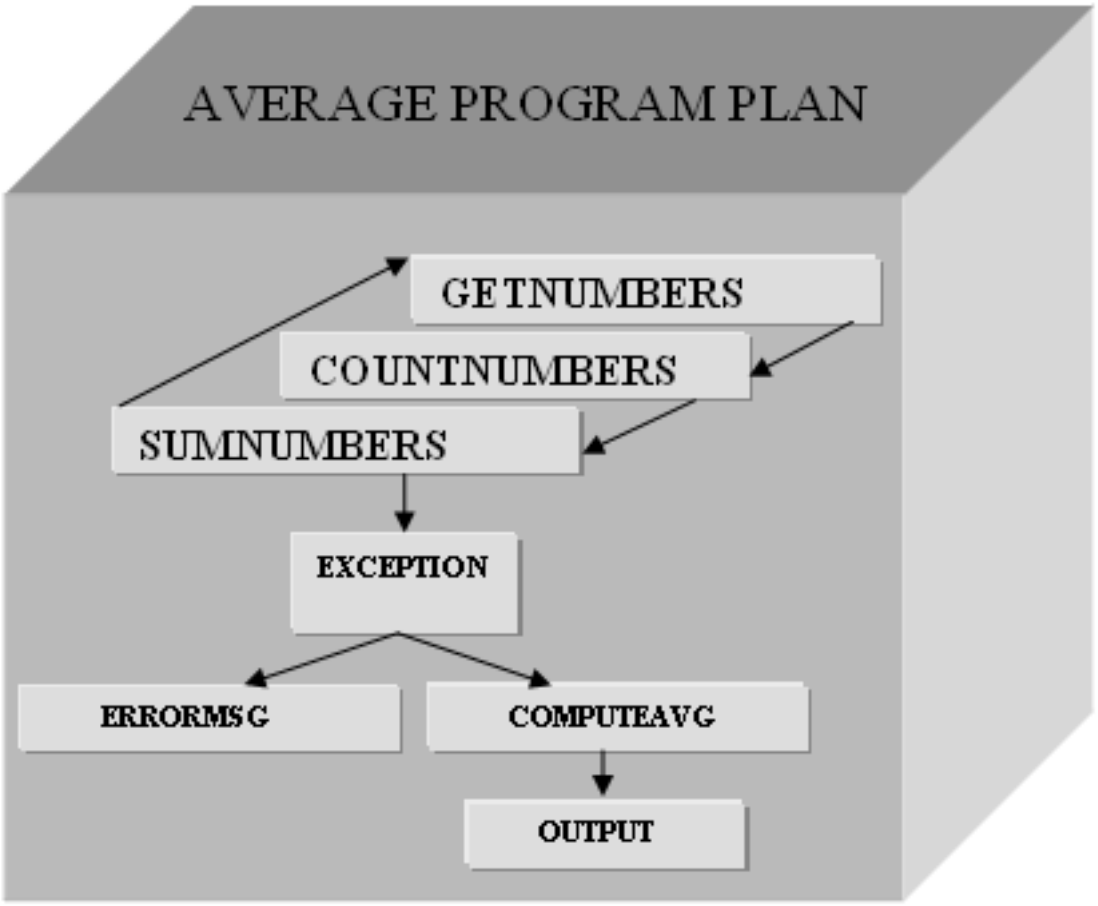

Figure 3 – Find Average Plan Example

The following steps demonstrate how plan integration is done for solving a problem. To find the average of a group of numbers, the find average plan could be decomposed into the following sub-plans: getnumbers, countnumbers, sumnumbers, exception, errormsg, computeavg, and output. In this example, the sumnumbers, countnumbers, and getnumbers plans are implemented so that they are interleaved with each other. This means that when a number is input (getnumbers), the counter is incremented (countnumbers), and this number is added to the sum (sumnumbers); now the process returns to the getnumbers plan and starts again. When all numbers are entered, the process follows sequentially to the exception plan (an appended plan), which checks for an error, such as division by zero. The errormsg and computeavg plans are branched plans, meaning that the errormsg plan will be entered on the condition that an error exists, otherwise the computeavg plan will be entered. The output plan is appended to the computeavg plan and outputs the result [Ebrahimi, 92].

Phase 3 of Learning: Plan Creation

A learner's skill can be measured by the ability to solve a problem entirely, from problem specification and requirements to plan decomposition to plan integration, and finally to testing the correctness of the plan and its explanation. In this phase, the learner is responsible to create a plan and demonstrate problem solving skills. The problem could be one that already exists in the WVLS library, or it could be a new problem. If the problem exists in the system, there are three incremental learning scenarios: 1) Full access - The total WVLS is at the service of the learner. The plan can be observed; the learner can be asked to integrate the plan, and the system will assist by indicating if the integration is right or wrong. 2) Limited access of WVLS- Access to observation of that particular plan is denied and only the plan integration phase is available. 3) Denied access to WVLS - The learner must decompose and integrate the plan without accessing the observation or integration phases for that plan. If the plan exists, the learner is encouraged to use the system resources to help devel-

op the plan, such as observing the plan and using existing sub-plans.

When a learner requests to create a new plan that is not part of the system library, the entire WVLS will be at the service of the learner, including all phases. However, the new plan consists of several sub-plans that already exist in the system and are shared by other plans. Shared sub-plans can be reused as they are or modified to suit the purpose. To assist the learner with creating a new plan, an intelligent system will monitor and profile the learner in the background, while the problem-solving steps are taking place. The intelligent system will try to determine the intention of the user based on collected clues and then make suggestions along the way.

Assess and Assist the Learner

In order to assess and assist the learner, the cause of the problem must be identified and reported to the learner as early as possible, preferably in a systematic or a standard way. During plan creation the cause of errors can be classified into four categories known as: missing, misplaced, malformed, or misconception (spurious). A plan is missing when it does not exist; a plan is misplaced when it is out of place; a plan is malformed when it is not complete or partially complete; a plan has misconception when it's totally wrong or has nothing to do with the problem specification. During plan integration, the learner is tested to see if the plan integration is understood. Given two plans at a time with four possible modes of integration such as: appended, interleaved, embedded, and branched, WVLS will report if the integration mode is incorrect. At each stage, the learner is given the suggestion/option to consult any prior phase.

Learning Theories and Techniques

Ertmer and Newby (1993) compare Behaviorism, Congnitivism, and Constructivism learning theories and suggest that an instructional designer should choose a different strategy for different learning situations. To serve a community of individuals with diverse learning backgrounds, WVLS integrates many different learning techniques as much as possible. The hope is that, if one learning method is not successful, eventually one of the other methods from the pool of learning techniques will be. Moreover, some learners understand things better when they are represented in different ways. As Wilson (1997) points out, instructional designers must be aware of the advantages and disadvantages of each learning theory so that they can optimize their use in proper instructional design strategy.

WVLS learning techniques include: learning by analogy (associating a new problem with a similar one), learning by chaining (new ideas are related to what the learner already knows), learning by doing (testing and observing problems being solved), learning by creating a demand and providing a supply (a new plan is introduced only when it is needed), and learning by error (solving a problem by trial and error). Bednar (1995) pointed out the importance of linking theory to practice in the design and development of any instructional system and emphasized that effective design is possible only if the designer is aware of the theoretical foundation underlying the design. WVLS design has taken the initiative to bring learning theories into practice as much as possible.

VPCL: A WVLS for Novice Programmers

The systematic way of breaking down a problem into plans and integrating them is demonstrated in the design of a system known as VPCL, Visual Plan Construct Language. VPCL is a tool for teaching beginner and novice programmers. Textual representation of programs and sub-programs (functions) contributes to misconceptions of how sub-programs interact with each other. Visual plan representation has resolved some of these problems. Plans are independent of programming functions. A plan can be a single statement like adding one to a counter, or a single concept; anything that is important or needs to be discussed can be a plan. It is emphasized that a function is a plan, but a plan is not necessarily a function. Functions (sub-programs) have restraints on how they are used and how they interact with each other. A plan does not have these restrictions. Plans can either be dependent or independent of the programming language used. One task of VPCL is to encourage the learner to think in terms of plans and sub-plans structurally by building abstraction on each level, while not overwhelming the learner with scattered knowledge. The idea of plan programming differently has been used as early as mid eighties; for example, Soloway uses the idea of Programming Plan to represent a typical action sequence in programming such as a loop to calculate a total or to search for an item. [Soloway, 1984] A novice programmer's support environment using plans can be created using hypertext to show the relationship between plans as well as breaking down the program to the smallest piece possible [Liffick, 1996].

The plan creation phase of VPCL deals with writing actual programming code. In this phase, the user is given problem specifications and is required to develop all the steps to arrive at a solution. To translate a problem into a written program, several plans will be needed. These plans may exist and need to be modified or may be created from scratch. After all necessary plans are created, the plans must be integrated, and then the program is executed.

Several learning strategies and systematic ways of evaluating the learner's problem solving skills are incorporated in VPCL. Intelligence is built into this system to try to figure out what the learner is trying to do. For example, if someone is writing a program that contains a comparison within a loop, the system will recognize this as a search program. The system can also store profiles of the users, so it will record the strengths and weaknesses of the learner. The effectiveness of VPCL as a learning and instructional tool has been shown by the result of empirical studies on novice programmers [Ebrahimi, 92].

Conclusion and Future Remarks

The advantage of web and visualization with three incremental phases of learning in WVLS ensures the understanding of the subject matter. Different levels of abstraction are applied at each phase in order to avoid overwhelming the learner or distracting the learner from reaching the main goal. WVLS promotes a standard way to communicate in decomposing a plan, integrating plans, and pinpointing the real cause of errors and building a new plan. The WVLS visualization and standardization bridges the gap of misconceptions. While WVLS has been tested on novice programmers, it is a generic environment that can be applied to other subject matters such as learning math or a natural language. Future work on WVLS can be done to enhance the intelligence of the system in reporting errors, assisting the learner, and profiling the learner more accurately.

Several empirical studies of novice programmers suggest that the understanding and productivity of programming can be improved by plan composition and visualization. Further empirical studies need to be conducted on learners of different subjects to determine the effectiveness of WVLS.

References

Arnow, D. and Barshay, O. (1999) WebToTeach: A Web-based Automated Program Checker. Frontiers in Education (FIE '99), San Juan, Puerto Rico (Nov., 1999).

Bednar, A.K., Cunningham, D., Duffy, T.M., Perry, J.P. (1995). Theory into practice: How do we link? In G.J. Anglin (Ed.), *Instructional technology: Past, present and future* . (2nd ed., pp. 100-111), Englewood, CO: Libraries Unlimited, Inc.

Conway, M., Pierce, J., Pausch, R., et al. (2000) Alice: Lessons Learned from Building a 3D System for Novices. *Proceedings of CHI 2000,* pages 486-493.

Ebrahimi, A. (2002) *C++ Programming Easy Ways,* American Press, Boston

Ebrahimi, A (1992) VPCL: A Visual Language for Teaching and Learning Programming (A Picture is Worth a Thousand Words). In *Journal of Visual Languages and Computing* 3, 299-317

Ertmer, P. A., Newby, T. J. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly,* 6 (4), 50-70.

Liffick, B., Aiken, R. (1996) A novice programmer's support environment. *Proceedings of the 1st conference on Integrating technology into computer science education, Volume 28, 24 Issue SI , 1-3* Niederst, J. (2001). *Web Design in a Nutshell (2/E)*. Sebastopol, CA: O'Reilly & Associates.

Schank, R. & Abelson, R. (1975). Scripts, plans and knowledge. Proceedings of the fourth international joint conference on artificial intelligence, Tbilisi. Re-published in P. Johnson-Laird and P. Wason, (1977) (Eds.), Thinking. Cambridge, England: Cambridge University Press.

Soloway, E. (1984). A Cognitively-Based Methodology for Designing Languages /Environments/Methodologies. *Proceedings of the ACM Symposium on Practical Software Development Environments.*Truong, N., Bancroft, P., Roe, P. (2005). Learning to Program through the Web. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education ITiCSE'05*

Wilson, B. G. (1997). Thoughts on theory in educational technology. *Educational Technology,* January-February, 22-27. *'05*

About the Author

Dr. Alireza Ebrahimi

Dr Alireza Ebrahimi is a Professor of Management Information Systems at State University of New York/ College at Old Westbury where he has taught programming for twenty years, also director of Teaching for Learning Center. He received a chancellor's award for teaching excellence in 1993. Developed a visual language known as VPCL (1989). Professor Ebrahimi has taught at other institutions such as Fordham University, NYU, Polytechnic University of New York, and Nassau Community College. He has also served as ACM lecturer, consultant / trainer to NYNEX and Honeywell. Dr Ebrahimi received all his three degrees in Computer Science and his graduate study was at NYU and Polytechnic University where he earned his Ph.D. in 1989.

THE INTERNATIONAL JOURNAL OF LEARNING

SCOPE AND CONCERNS
Visit: http://www.Learning-Journal.com

SUBMISSION GUIDELINES
Visit: http://www.Learning-Journal.com

INQUIRIES
Email: cg-support@commongroundpublishing.com