Heterogeneous Information Resources and Asynchronous Workgroups: Creating a Focus on Information Discovery and Integration in Computer Science

Wendy A. Lawrence-Fowler & Richard H. Fowler
Department of Computer Science
University of Texas - Pan American
Edinburg, TX 78539-2999

Abstract - *The widespread availability of network communication allows the educational community to provide students with skills needed for the pervasive changes in information access and utilization that are occurring. These skills center on developing techniques for learning in an environment which has a broad range of heterogeneous information resources. The university environment we create provides students with opportunities to acquire skills needed for success in a distributed workplace in which electronic tool use and information discovery, organization, and utilization are the principal activities in problem solving.*

Introduction

This paper reports on an approach to teaching in an introductory programming course. CS 1380: An Introduction to Computer Science is a standard introductory computer science course which provides students both a theoretical foundation for continued study and applied skills to begin implementing programs. Students must learn general problem solving skills, algorithm design and a programming language. Moreover, software engineering and specific program design techniques are learned from the outset. Maintaining balance and perspective among these somewhat disparate skills is one of the principal challenges. A second challenge lies in preparing students to apply these skills in a workplace inundated with heterogeneous information resources and tools to mange the resources.

Our reasoning is that if diverse skills are required and if the work environment has a broad range of heterogeneous information resources, students should have the opportunity to acquire the skills necessary for success in a distributed workplace where electronic tool use and information management are the principal activities in problem solving. Providing an intensive electronic environment and structuring student activities through electronic communications creates small scale communities within which students work. A persistent focus on information discovery and integration is maintained. Tools that coordinate information access, pedagogy, problem solving and design are at the core of the student's learning and working environment.

The Programming Design and Programming (PDP) environment we create combines networks, workstations, student laboratories, a pedagogic CASE tool (Design Tool), a programming environment, and web-based hypermedia as instructional tools. Course content is tightly coupled through a WWW based hypertext of lecture materials, lab exercises, and example programs [5]. The hypertext contains text and static graphics as well as animations to illustrate control and data structures through program execution. Students access the WWW materials and use the visually based Design Tool and programming language in program design and implementation. Asynchronous workgroups are used in team software engineering projects. The projects are facilitated through a local newsgroup which serves as a repository for group code and reports.

Activities and Design Goals For A Learning Environment

The PDP environment is designed to foster a general orientation to learning that is self-directed, active and group-oriented. General activities and types of interaction addressed by the environment include work community relations, electronic tool use, information access and utilization, and efficient use of human resources. An expansion of these activities is presented below in table 1. The elements of the list are drawn from discussions of the future of the workplace and society [2,3,8], and other's experiences in implementing such learning environments [1,4,9]. Additionally, members of the computing industry, as early adopters of information technologies in the workplace, have provided important insights into the activities and interactions of present and future information workers. Finally, our own experience of several years in using hypermedia in our classes [6] and the unique needs of our student population have shaped the activities and design goals.

Designing and implementing a university learning environment which addresses the complete set of design goals and activities outlined makes it necessary to create a distributed work environment which promotes interaction among students.

0-7803-4086-8 ©1997 IEEE

Integrating Networks, Workstations, Student Laboratories, and Multimedia

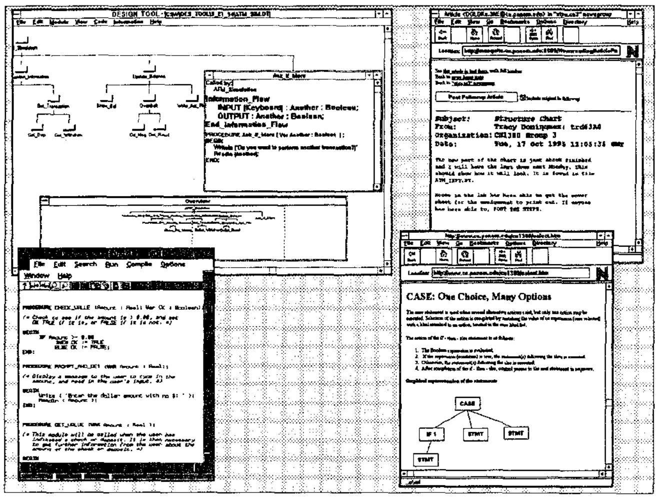
The PDP learning environment utilizes networks, workstations, student laboratories and multimedia to prepare students for self-directed learning in a distributed work environment. Structuring of student activities through laboratories and electronic communication is designed to create small-scale communities within which students work. These elements are used as part of a computer science course focusing on software engineering techniques for program development. Given course content and industry practice, the extensive use of electronic tools for software development is natural and the cornerstone of the course.

Table 1. Activities supported to provide a university learning environment preparing students for self-directed learning in an information rich, distributed work environment

Work Community Relationships	• new communities and organizations • synchronous and asynchronous communication • electronic communication within and between group members • changing roles from learner to mentor to collaborator
Electronic Tool Use	• pervasive use of electronic tools in all aspects of work • integrated suites of tools with diverse functionality
Information Access and Utilization	• facile in manipulating diverse information resources • iterative searching and collating of information • information reuse across time and domains • joint information use
Efficient Use of Human Resources	• recognition of individual learning styles • support multiple mechanisms for learning and working • provide multiple sources of information

The environment provides students with an integrated suite of pedagogically oriented electronic tools and heterogeneous information resources. The tools and the interrelationships among the tools are designed to support independent development of each of the skills necessary for programming and managing information resources, while keeping more abstract concepts of computer science in sight.

We have integrated Information sources and tools that support students in the distributed environment including WWW and ftp sites with instructional materials (hypertext, user defined indices, program designs, algorithms, and sample programs), web browsers, news groups, and e-mail. Figure 1 shows a student's screen when working in the distributed environment. Four windows are open on the screen. The upper left window contains the student CASE (computer assisted software engineering) tool displaying a programming team's design for a project. The complete design is visible in the smaller window within the CASE tool window. The Pascal editor is in the lower left window. Information describing a team member's progress, reported via a project newsgroup, is viewed using a news reader in the upper right window. Hypertext explaining a programming construct is stored at the course's web site and is viewed through a web browser open in the lower right window. Whether local or networked, text can be transferred between the information access components (web browser and news reader) and the programming tools (student CASE tool and programming environment).


DESIGN TOOL
File Edit Module View Code Information Help
Overview
File Edit Search Run Compile Options
Window Help
File Edit View Go Bookmarks Options Directory Help
Back Home Reload Open Find
Location:
Post Followup Article
CASE
IF 1
STMT
STMT
STMT

Ask_If_More

```
Called by:
   ATM_Simulation
Information_Flow
   INPUT [Keyboard] : Another : Boolean;
   OUTPUT : Another : Boolean;
End_Information_Flow
PROCEDURE Ask_If_More ( Var Another : Boolean );
BEGIN
   Writeln ('Do you want to perform another transaction?')
   Readln (Another);
END;
```

```
PROCEDURE CHECK_VALUE (Amount : Real; Var OK : Boolean);
(* Check to see if the amount is > 0.00, and set
   OK TRUE if it is, or FALSE if it is not. *)
BEGIN
   IF Amount >= 0.00
      THEN OK := TRUE
      ELSE OK := FALSE;
END;

PROCEDURE PROMPT_AND_GET (VAR Amount : Real);
(* Display a message to the user to type in the
   amount, and read in the user's input. *)
BEGIN
   Write ( 'Enter the dollar amount with no $: ' );
   Readln ( Amount );
END;

PROCEDURE GET_VALUE (VAR Amount : Real );
(* This module will be called when the user has
   indicated a check or deposit. It is then necessary
   to get further information from the user about the
   amount of the check or deposit. *)
BEGIN
```

```
Subject:       Structure Chart
From:          Tracy Dominguez: [illegible]
Organization:  [illegible] Group 3
Date:          Tue, 17 Oct 1995 11:05:3[illegible]
```

CASE: One Choice, Many Options

Figure 1. Sample student screen while working in the PDP environment.

Laboratory

The course's laboratory component provides the unique benefit of bringing people together in the same place and at the same time. Whether lab work is completed independently or as a group, the opportunity exists for students to engage in informal discussions with faculty, lab assistants and peers, gaining new perspectives. The lab affords students the additional opportunity for systematic face to face, or synchronous, communication and

0-7803-4086-8 ©1997 IEEE

complements the asynchronous nature of the students group projects. Group and project management lab exercises help students learn about collaboration as well as learn using collaboration. Students gain experience in directly coordinating efforts with others whose skills and knowledge may differ greatly from their own. Lab interactions support subsequent collaborative activities by providing a foundation for the subsequent coordination of efforts through newsgroups which is necessary to complete group projects.

Structuring of student projects is done with care. Students are typically in their first or second year at the university, and this is often their first extensive experience with group projects. Initial group formation and project definition is completed during scheduled laboratories. Based on student performance and previous projects, groups are systematically formed and reformed several times throughout the semester. Individual students are given the opportunity to fulfill both mentor and learner roles within groups.

Electronic Communications: Newsgroups and E-mail

Local newsgroups are used to foster the emergence of new communities. In this case, the communities are composed of computer science learners. At the outset of the course, a single course-wide newsgroup is used for communication concerning work assignments and student commentary. Students are not only encouraged, but required to contribute to ongoing, monitored, but unmediated discussions. The newsgroups provide an element of immediacy, or at least relevance. The well known synergy of newsgroups does develop. Students use the newsgroup to share ideas and information about meetings, to raise questions, to respond to questions, and to rebut or refine another student's response. Lines of communication open as student responses trigger further discussion.

The archival features of the newsgroup allow the electronic conversations to persist through time. This supports the formation of a community by eliminating the temporal constraint of face to face conversation. Ours is a commuter campus and these newsgroups are a significant factor in creating a computer science community. Additional project oriented newsgroups are created during the semester and used for communication about team projects. These newsgroups work well because they support asynchronous work, better suited to work patterns of students at a commuter campus. Finally, instructors can monitor individual student involvement and group progress in a convenient way as compared to group communications conducted exclusively via e-mail.

E-mail is reserved primarily for student communication with the instructor. However, students can choose to post correspondence to a newsgroup if judged useful to a larger audience. The use of e-mail for instructor-student interaction has provided a different structuring of the instructor-student relation. It overcomes some of the practical difficulties of providing rapid feedback on questions. Moreover, students are relatively thoughtful in framing their questions. Their messages tend to be brief, more focused, and directed to a single or limited number of topics. In turn, faculty respond to the student at the appropriate skill and knowledge level.

A CASE Tool for Pedagogy: Design Tool

Design Tool provides a visually based system for problem solving using problem decomposition, program design via structure charts, data flow checking, Pascal code generation, and report production. Design Tool provides general tree display facilities for module creation and deletion, reordering and repositioning of modules, collapsing and expanding subtrees and windows, and various layouts of the tree structure [5]. As shown above in Figure 1, an overview diagram of the entire structure can be used in orienting the displayed modules to the complete program design. In addition to providing an orientation to the complete project design, the overview provides navigation facilities for moving within the design's module structure. The Display facilities for panning, using the overview, and zooming in and out on parts of the structure are intended to help the students keep track of the overall design during development.

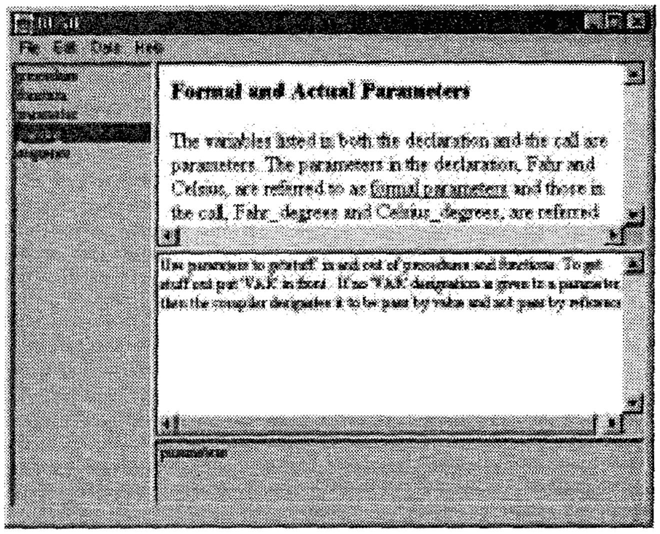
Hypertext Notebook

The web-based hypertext serves to integrate concepts introduced in the classroom and text with the design and programming environment. The notebook contains the course lecture material with static graphics, animations illustrating control and data structures, and course supplements including course syllabus, schedule, laboratory exercises, programming assignments, and a large set of designs and example programs.

The hypertext is structured into modules which reflect the sequence and scope of lectures. Within modules, main lecture topics serve as organization points from which students can explore concepts in greater depth. Each module provides numerous examples illustrating newly introduced concepts that build on previous examples and concepts. Students can experiment by copying programs into the programming environment, compiling, and executing the code. All lecture examples are included in the hypertext. Thus, students can focus on discussion in the classroom, later using the hypertext to review examples.

Student Defined Index: DUSIE

Tool intensive and highly integrated, the PDP virtual classroom provides a rich assortment of resources to support information retrieval and understanding. However, the original tool set, discussed above, did not include a tool which facilitated the capture and organization of meta-knowledge about the resources, particularly the hypertext documents. We recently developed an indexing tool, based on the idea that it is the user's perception of the concept structure and the ability for users to organize their concepts as they perceive them that is important in the learning environment. The Dynamic User-defined Searchable Index Engine (DUSIE) allows the student to construct a personal repository of concepts and concept relations to complement the author centered hypertext document structure [7]. As an extension of the virtual classroom, the tool reinforces learning through the active construction of an associative network of terms and annotations. As shown in figure 2, students create an index into the hypertext document which explicitly defines their understanding of the concepts. They impose their ordering on the concepts and modify their ordering as their perceptions of the concepts change.



Formal and Actual Parameters

The variables listed in both the declaration and the call are parameters. The parameters in the declaration, Fahr and Celsius, are referred to as formal parameters and those in the call, Fahr_degrees and Celsius_degrees, are referred

Use parameters to get stuff in and out of procedures and functions. To get stuff out put 'VAR' in front. If no 'VAR' designation is given to a parameter then the compiler designates it to be pass by value and not pass by reference

parameter

Figure 2. Sample DUSIE screen displaying the web-based hypertext, annotation and related terms list for the index term passing.

In the simplest terms, DUSIE provides users facilities to construct an index for Web documents. Users define a list of index terms, make annotations to the terms, and link related concepts. In the context in which DUSIE was developed, a primary function of the indexing process is to provide a set of terms and personal annotations for each document that reflects the student's conceptual domain and can be used efficiently and effectively to retrieve information from the documents. More generally, the system's functionality supplies mechanisms to create 1) new nodes through the annotation facility, 2) links to points in existing Web documents for the new nodes, and 3) alternative links for existing Web nodes through the addition of terms linked to points in existing documents.

Discussion

Our goal in developing the PDP environment is to provide an environment which allows students to develop the required skills to manage information and solve problems in an environment analogous to that of the present and future workplace. The PDP system is designed for introductory students, therefore it is critical that they gain an understanding about what and why information is significant. It is important that they learn how, when, and where to apply knowledge.

The range of tools used in the course integrate information access, pedagogy, and program development. These electronic tools are at the core of the student's learning and working environment and require the student to master several tools using different forms of information: the web browser accessing course hypermedia (information and problem statements in natural language), the CASE tool (formal program specification), the programming environment (Pascal language), the news groups (group progress and public commentary), and e-mail (personal communication). As with many "paperless courses", assignments and lecture content are available on-line. In our implementation this material is available on the Internet, and students can access it off-campus. While not encouraged to do so, the student is able to complete most aspects of the course off-campus using the online course materials together with newsgroups and e-mail.

It is the interaction among these tools and the transfer and transformation of information among these tools which creates the experience of working in an information intensive environment. Students become familiar with and facile in the transfer and translation of one form of information to another. In addition to information resources provided for the student, the stored problem solutions and code in the hypermedia form a repository which is systematically used throughout the course. The skill and care with which students create and utilize these information elements impacts success in projects which reuse information created earlier in the course. In essence, students learn to use tools and manage information, facilitating the process of scholarship.

As an artifact of the environment design, both linear and exploratory styles of learning, and thus, the reported differential learning strategies employed by higher and lower ability students are supported [10]. Higher ability students can explore the whole system interactively while the student who is overwhelmed by the information can follow a more linear path through the information.

Conclusions

Widely accessible technologies and tools are available to implement a university learning environment which trains students for an information-intensive workplace. Though some elements of the environment we have implemented for our computer science classes are specific to the discipline, most of the elements might be used in any domain. For example, e-mail, publicly available news readers, and web browsers have been successful in creating communities centered in the course. Even where the particular tools used in this project are not appropriate, analogs to those tools which transform and integrate information exist for virtually all disciplines.

However, it is important to remember that information technology is the medium to enable changes in curriculum and not the curricular content itself. Technology supports active learning by facilitating work on complex projects, rethinking assumptions, and discussion. Technology supports the development of life-long learners.

In this project we use technology to implement educational strategies which include collaborative, project based learning in an information-rich, tool-rich environment where both synchronous and asynchronous communication occurs to improve student-faculty and student-student interaction. Various styles and preferences of student learning are supported. We believe that these strategies can influence the course of a student's development and better prepare them for entry into the workplace of tomorrow.

The overall experience is valuable for the students. The students gain insight and skills that they might not otherwise acquire. Learning is enhanced because it is immediately supported by practical, hands-on application of theory and guidance by faculty, lab assistants, and peers. Students not only learn about collaboration, they learn through collaboration. As a step in building a teaching and learning paradigm which includes collaboration, this experience indicates that it can be a successful model.

Acknowledgments

A thank you must be extended to Xiannong Meng for contributing his theoretical and practical knowledge of distributed systems to this project and to Isidoro Pulido for his programming efforts.

References

(1) Andriole, S. J. (1995). Asynchronous learning networks: Drexel's experience. *THE Journal, 23(3)*, 97-101.

(2) Bates, A. W. (1993). Educational aspects of the telecommunications revolution. In Davies, G., & Samways, B. (Eds.). *Teleteaching*, North Holland, Amsterdam, 1-10.

(3) Boshier, R. (1980). *Towards a learning society.* Learning Press, Vancouver.

(4) Debreceny, R., Ellis, A., & Chua, K. (1995). The integration of networked learning delivery - from strategy to implementation. *Educational Multimedia and Hypermedia 1995*, Association for the Advancement of Computing in Education. Charlottesville, VA. 169-174.

(5) Fowler, W. A. L. & Fowler, R. H. (1993). A hypertext-based approach to computer science education unifying programming principles. *Journal of Educational Multimedia and Hypermedia, 2(4)*, 433-441.

(6) Fowler, W.A.L., & Fowler, R.H. (1996). Networks, workstations, multimedia, and electronic communities: Creating a university learning environment. *Proceedings of ED-TELECOM, World Conference on Educational Telecommunications 1996*, Association for the Advancement of Computing in Education. Charlottesville, VA. 103-108.

(7) Fowler, W.A.L., Fowler, R.H., Williams, J.L., Palacios, J.X.R., & Palacios, J.X. (in press, June 1997). DUSIE: Augmenting a static web with user defined content and link structure. *Proceedings of ED-MEDIA, World Conference on Educational Media 1997.* Association for the Advancement of Computing in Education. Charlottesville, VA.

(8) Hiltz, S. (1984). *Online communities: A case study of the office of the future.* Ablex, New Jersey.

(9) Norrie, D. H. & Gaines, B. R. (1995). The Learning Web: An agent-oriented network supporting knowledge access, collaboration and simulation. *Educational Multimedia and Hypermedia 1995*, Association for the Advancement of Computing in Education. Charlottesville, VA. 500-505.

(10) Reckert, M. & Pirolli, P. (1992). Student strategies for learning programming form a computational environment. 2^{nd} *International Conference on Intelligent Tutoring Systems.* Springer-Verlag, Berlin. 382-394.