The Lab and the Web: Transforming the Sophomore Experience

W. M. Waite, R. Simpson
University of Colorado at Boulder/Eastgate Systems

ABSTRACT: In the fall of 1995 we initiated a major revision of our sophomore course on computer architecture and assembly language in order to increase student involvement and provide more design experience. Its new title is "Computers as Components", and it uses embedded systems to motivate the necessary skill acquisition. This paper discusses the structure and support of this course, and our experience with teaching it.

1. Background

Falling enrollments and problems with retention of sophomores prompted us to examine our curriculum in 1992. We interviewed students, looked at initiatives at other schools, consulted with industry representatives, and debated strategy and tactics internally. Our conclusion was that we needed to improve the students' laboratory experience and integrate it more closely with lecture material. In that way, we felt that we could provide stronger motivation for the lecture material and also reinforce it through immediate application.

There has been a trend at the University of Colorado towards a separation of lectures from laboratories. A number of arguments have been put forth in favor of this separation, probably the most understandable being that of scheduling flexibility. Unfortunately, this separation leads to a psychological distance in the minds of both the faculty and the students. Even when a lecture and a laboratory are co-requisites, they are distinct. Since scheduling flexibility is an important consideration, different faculty members will usually be responsible for them. They are graded separately, and there is little incentive or opportunity to try to integrate the material closely. Thus much of the benefit of presenting the same material in different ways, exercising different learning styles, is lost.

We believed that this trend needed to be reversed, so we postulated a set of core courses that would integrate lecture, recitation and laboratory. Most of the courses were already in place and required, but were offered as separate lectures and laboratories. The existing lectures carried three credit hours and the existing laboratories carried one, so in order to accommodate a recitation we had to increase the total number of credit hours per course to five. Because of the inflexibility of the University's scheduling program, we were forced to offer these courses as though they consisted of three one-hour lecture sessions and two two-hour laboratory sessions per week. This turned out to be a blessing in disguise, because it allows considerable flexibility in how we use laboratory sessions.

In addition to converting to the new format, the digital faculty took this opportunity to re-order our two offerings at the sophomore level. All freshmen in the College of Engineering take (or place out of) an elementary programming course. Prior to the fall semester of 1995, sophomores in electrical engineering, computer engineering, and computer science then took a course in logic circuits followed by a course in assembly language programming. Our experience was that virtually none of the material from logic circuits was ever used in the subsequent course, and that there was very little motivation for the students to learn it.

We therefore decided to reverse the order of these two courses in the process of changing them to the 5-hour format.

Finally, since almost every system in any field of engineering involves embedded computers, we tried to broaden the erstwhile assembly language programming course into one that would benefit engineers from other areas. Partially for this reason, but also to focus our attention on essentials, we changed its name and defined the three courses in terms of outcomes:

Elementary Programming: An understanding of the concepts of programming; familiarity with one imperative programming language; an ability to cast problem solutions in terms of algorithms.

Computers as Components: An understanding of one machine architecture and its associated assembly language; familiarity with the use of computers as system components; an ability to cast problem solutions in terms of a mix of hardware and software.

Digital Electronics: An understanding of common digital logic components and subsystems: familiarity with the techniques of digital design; an ability to cast problem solutions in terms of hardware.

This, then, was the starting point for the design and implementation of "Computers as Components" in the 5-hour integrated format. We will look at two-specific aspects of the design, our use of the laboratory and our use of the World-Wide Web, and then try to draw some conclusions from our experiences.

2. Use of the Laboratory

We wanted to truly integrate the laboratory with the course, and also to use the laboratory to encourage students to work together in problem solving and studying. The latter was important both because our industrial customers want more team experience and because we believed that students could learn effectively from each other.

All of the homework for the course was associated with the laboratory. Each assignment was divided into two parts, the first of which required the students to answer questions related to the homework topic and/or provide a design for a problem solution. The second part of the assignment required them to implement and demonstrate their solution.

2.1. Mechanics

Teamwork was required for all assignments, with the team handing in one set of solutions and everyone getting the same grade. We did not accept individual assignments under any circumstances, but homework grades only made up 20% of the total grade for the course. (There were also two 50-minute examinations worth 20% each and a three-hour final worth 40%.) Thus we encouraged and rewarded effective teamwork, but still had sufficient individual information that it was impossible for a student to ride through solely on the work of their teammates.

There has been considerable debate in courses involving teamwork about how the teams should be formed. We believe that team formation and maintenance should be the responsibility of the students. If the professor assigns the teams, then any problem is the professor's problem; if the students form the teams then they also own the problems and must solve them. They must, of course, be given help in recognizing and solving those problems.

The team was responsible for preparing appropriate demonstrations of their practical work. For each assignment, they needed to develop a demonstration lasting no more than five minutes that verified they had completed the assignment and showed that they understood the principle the assignment illustrated. After the conclusion of the demonstration, the TA asked questions about the assignment. Demonstrations were graded on the basis of clarity (50%) and demonstrated understanding (50970). Each demonstration was given by a single team member, who then had to answer all of the TA's questions. (The rest of the team needed to be present, but they could not participate.) The grade for the demonstration was then recorded for every member of the team, and thus the team's success depended upon the success of the demonstrator. We did not allow the same person to act as the demonstrator in successive weeks, and required that this role be evenly divided among the team members over the course of the semester.

This approach to demonstrations has two benefits: It requires the team to distill their week's work into a convincing five-minute presentation, and it forces them to coach and rehearse a single spokesperson to make that presentation effectively. We emphasized in our instructional material that the ability to effectively present their work was vital to their success in engineering, and that such presentations were almost invariably made under a time constraint. We expected that this aspect of the grading would be a real point of contention. but that was not the case.

2.2. Assignments

The platform we used for the lab was a PC with an off-the-shelf interface card that provided 16 analog inputs, two analog outputs, 8 digital inputs, 8 digital outputs, and 3 counter/timers. Our software was MS-DOS, Windows 3.1, Borland C, Netscape, and shareware for TCP/IP communication. With the exception of the interface board, this is the computing environment provided in many of the open computing labs run by our Computing and Network Services group. Thus students could use CNS labs for program preparation and initial debugging, relieving much of the pressure on our machines.

The first assignment simply familiarized the students with the computing environment. It involved obtaining source code from the World-Wide Web, compiling and using that code, and using electronic mail to obtain data and submit results. The next group of assignments were concerned with various aspects of assembly language, and communication between code written in C and code written in assembly language. Assignments involving programmed control of several 1/0 devices were next, followed by interrupt-driven 1/0 operations. Finally there were assignments in which several devices and techniques had to be integrated.

Students were given progressively more design responsibility during the semester, with the first assignments being very tightly specified and the later ones quite loose. In all cases, however, the overall shape of the assignment was given. We believe that freely-chosen projects would not be compatible with the goals of the course.

All of the assignments involved problems that were as realistic as we could make them within the scope of the time and equipment we had available. We tried to make each illustrate not only a particular technique or piece of hardware, but also a reasonable application for an embedded system. We believe that such problems are necessary to really engage the student's attention and convince them that the material they are learning is useful.

2.3. Maintenance

In order to present courses like "Computers as Components" effectively, we need to have the students write interrupt handlers, device drivers, and other low-level code. This mitigates against the use of a standard, protected operating system and thus causes serious problems in maintaining the integrity of the laboratory machines. One of the most difficult tasks that we faced in this course was keeping our stations running. We had the capability for downloading complete disk contents, but our procedures were too cumbersome and the network was too slow to make it a routine matter.

Programming errors in a group's solution to a homework exercise can easily corrupt the memory-resident operating system, but damage to disk files is less likely. Thus, most problems can be avoided if a group boots the machine at the start of every session. Since many of the exercises involve external hardware, however, it is not sufficient to have the computer itself in a known state. We had carefully described the state of any external hardware in the homework itself or in a hardware-specific writeup to which the assignment was linked, but it turned out that both we and they made too many assumptions. (For example, a group set up the configuration of a remote terminal very carefully but neglected to verify that it was connected to the computer.) We are now producing equipment verification programs that will provide end-to-end checks of specific external items; the associated documentation will suggest step-by-step troubleshooting procedures.

Students come to the class with a wide variety of backgrounds. Those who are sophisticated in the use of the PC want to operate in an environment as similar as possible to the one they are used to. They are quite happy to modify system configuration files in order to achieve that goal, thus confusing students with less experience and sometimes even bringing a system down. When they do make a mistake, they seem to be afraid to ask a TA for help; instead they quietly move to another station. We tried to combat this problem by requesting that students not make changes in the configuration, but that doesn't seem to work. We are now exploring methods of using cryptographic checksums on configuration files, and providing a selective download facility to repair localized damage.

Identification and reporting of inoperative setups is a vital task that we are not yet handling well. The problem is really one of convenience: A student comes to the lab, tries a setup, and can't get it to work. Since the lab is open continuously (with magnetic card access), and is generally unsupervised, it's hard for the student to tell anyone. Sometimes they put a note on the machine, but more often they just go to another setup. During a scheduled lab period the TA is kept busy answering students' questions, grading their demonstrations, and so forth. Again, inoperative setups are simply bypassed. We need to devise a trivial procedure that a user can carry out to indicate a failure electronically. Of course this procedure must work in almost all of the failure states . . .

3. Use of the Web

We wanted to use the World-Wide Web as a single source for all of the material related to the course, and as a mechanism to provide coherent guidance through its various components. The former application is well understood: any laboratory-based course requires a large amount of supporting material in the form of descriptions of equipment, laboratory procedures, and experiments. In addition, when the experiments involve computers, software must be made available. It is useful to have any software that is not part of the underlying system accessible from anywhere on campus, and to allow students to obtain copies of course material as the need arises. Using the web as a repository also simplifies dissemination of late-breaking news about laboratory status and problem workarounds.

In this section we will concern ourselves with the latter application, use of the web to provide coherent guidance through the components of the course. It is this application that must be addressed if we are to make use of computer technology to enhance student understanding. We first cover our basic strategy for providing orientation and support for differing student needs: reading the textbook, doing homework, and relating material to overall course goals and requirements. We then look at a particular mechanism to integrate all the themes of the course.

3.1. The Basic Strategy

We felt that our material needed to provide orientation for the student in both the time and the topic domains. Figure 1 summarizes the individual documents that were available to the students, and shows how they are related. "Web" refers to material that already existed on the web, to which we referred in the expanded textbook index and the structure maps; "paper textbook" refers to "Microcomputer Systems: The 8086/8088 Family", by Liu and Gibson (Second Edition, Prentice-Hall, 1986, ISBN O-13-580499-X). We created all of the other material mentioned in Figure 1 specially for this course.

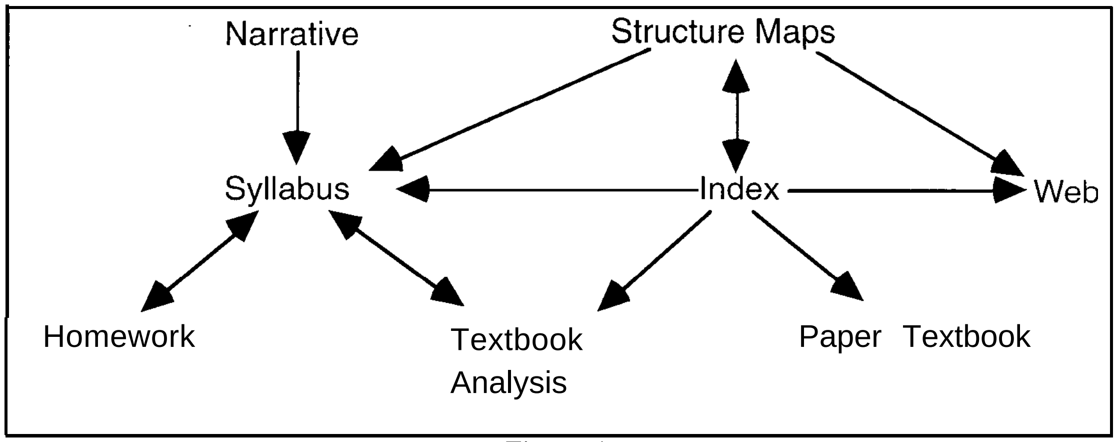


Figure 1
Relationships Among Major Documents

The course is designed to provide the necessary foundation for work with embedded systems, a foundation that includes machine architecture, assembly language programming, 1/0 device interfaces, etc. To integrate the detailed topics to be addressed in lectures and worked on in the laboratory, we created a realistic narrative that incorporated all of the material. This narrative context helped to motivate early mechanistic but necessary exercises by anticipating where they were going, and to ground later, more challenging exercises by providing the frame of reference for the whole.

The narrative line parallels the time line of the syllabus and contains links to each week of the syllabus; it provides orientation primarily in the time domain, giving an overview of the order in which the topics are to be covered and why that order makes sense. The narrative also explains which part of the story is covered by which exam. Thus, students not only had the syllabus topics list, but also the integrating narrative chunks to guide them in preparing for exams.

The course is top-down in its orientation while the textbook is a bottom-up, excruciatingly detailed reference. An additional complication is that the textbook index is truly terrible, making the textbook even more intractable than necessary.

Our first approach to the structure clash between the textbook and the course was to map the relevant textbook sections to the syllabus through a systematic section-by-section analysis ("Textbook Analysis" in Figure 1). Since some sections of the book were irrelevant to the course, and since the organization of the book bore no resemblance to the organization of the course, students attempting to read and absorb the entire book would be at a great disadvantage: not only would they waste time, but they would also lose the perspective that we were attempting to convey in the course. To forestall this, the textbook analysis indicated what was important, what was optional, and what was to be totally ignored in each section. The syllabus provided links to the relevant textbook analysis sections and the textbook analysis in turn linked back to the relevant syllabus descriptions.

Second, we decided to provide a thorough index to the textbook that linked into the course materials as well as providing much needed access to the book itself. This index contained 4860 entries and, when printed, was 70 full-size pages long (the original index for the 660-page book consisted of 10 half-size pages). The online index provided an excellent opportunity for topical guidance not only through the course material but also to other courses in the curriculum and to research on the web.

The mechanisms described in this section provide both time and topic orientation and support. The narrative is a time-based overview, while the textbook analysis and index give individual topic direction. The syllabus ties the two together with its lecture-by-lecture listing. Unfortunately these are all "large" documents that one cannot easily grasp in their entirety. What we felt was missing was a graphical topic overview (semantic map) that would integrate all the themes of the course, and this is the subject of the next section.

3.2. Structure Maps and Link Spaces

Guidance in the topic domain is most easily provided graphically, by a two-dimensional hierarchical structure that shows the relationships among topics at a particular level. Unfortunately, the very simple hypertextual foundations of the Web provide no mechanisms for creating a dynamic graphical overview. In addition, the Web gives no guidance as to the meaning or context of a link and allows only one-to-one linking, which is a serious disadvantage when trying to show a semantic map over a domain.

To solve this problem, we used Eastgate Systems' Story space (tin) hypertext authoring system' to create a fully-linked hypertext for the four major course themes. We then converted this hypertext into HTML imagemaps (structure *maps*) and text files (link *spaces*). The structure maps provide the graphical overview, while the link spaces embody both one-to-many links and descriptive material about those links.

The final semantic map contained 17 hierarchically-organized structure maps and 218 associated link spaces. The link spaces interconnect the structure maps into a generalized net hypertext as well as linking them into the rest of the course materials, other courses in the curriculum, and research areas on the Web.

Figure 2 shows the top level structure map for the course. Each node is represented by a rectangular element that has two parts. The lower part identifies the node concept and upper part gives the relationship of that concept to the node that links to it (for example, System Architecture is a key theme of this course). The upper part of a node is always active, and clicking on it displays the associated link space. The lower part is active only if there is a structure map for which this node concept is the central theme. In Figure 2, the lower parts of System Design, System Architecture, Software, and Embedded Systems are all active. Clicking on the lower part of any of these nodes displays the associated structure map.

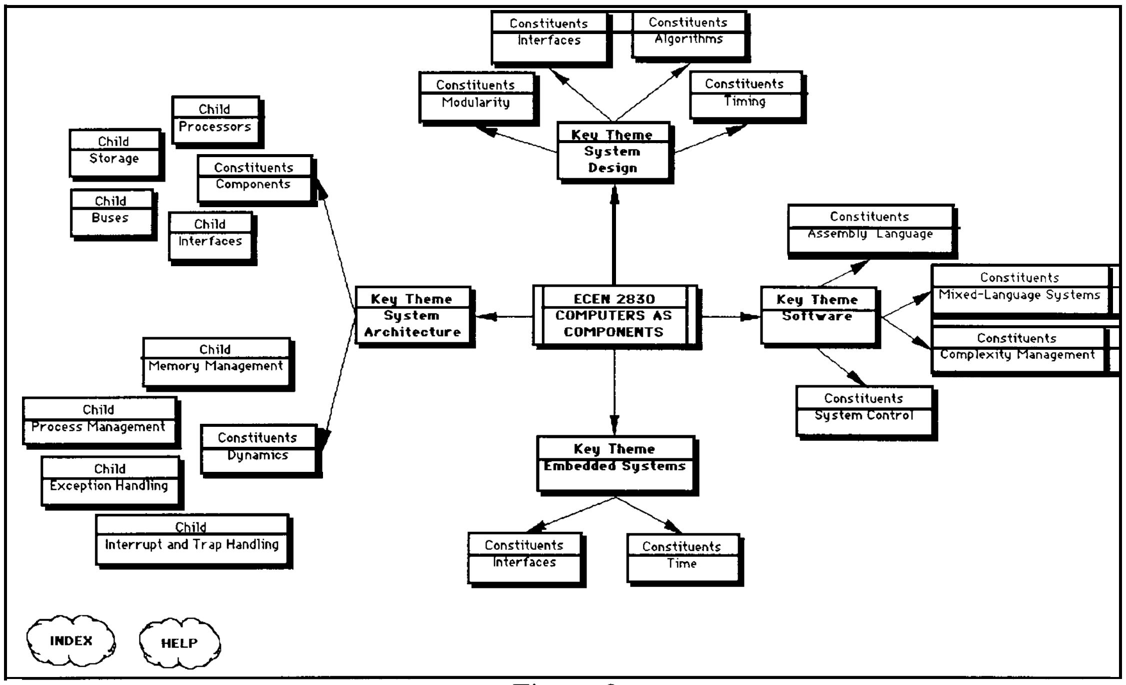


Figure 2
The Top-Level Structure Map

Two mechanisms for displaying hypertextual relationships are combined in the structure map: traditional node-link connectors 'and spatial proximity[2],[3]. (Spatial proximity is used here as a graphic-al device that reduces visual clutter; the underlying linking mechanism is through the link spaces.) The index and help buttons are always located in the lower left corner of the structure map to provide a consistent help and index access interface.

Figure 3 shows the structure map of a lower level in the System Architecture key theme guided trail[4]. The upper right-hand corner provides context through a modified fisheye view[5] of the course. The trimming of detail allows a clear view of the relationship between the current level and the rest of the structure. Each node, including all those in the fisheye view, is active.

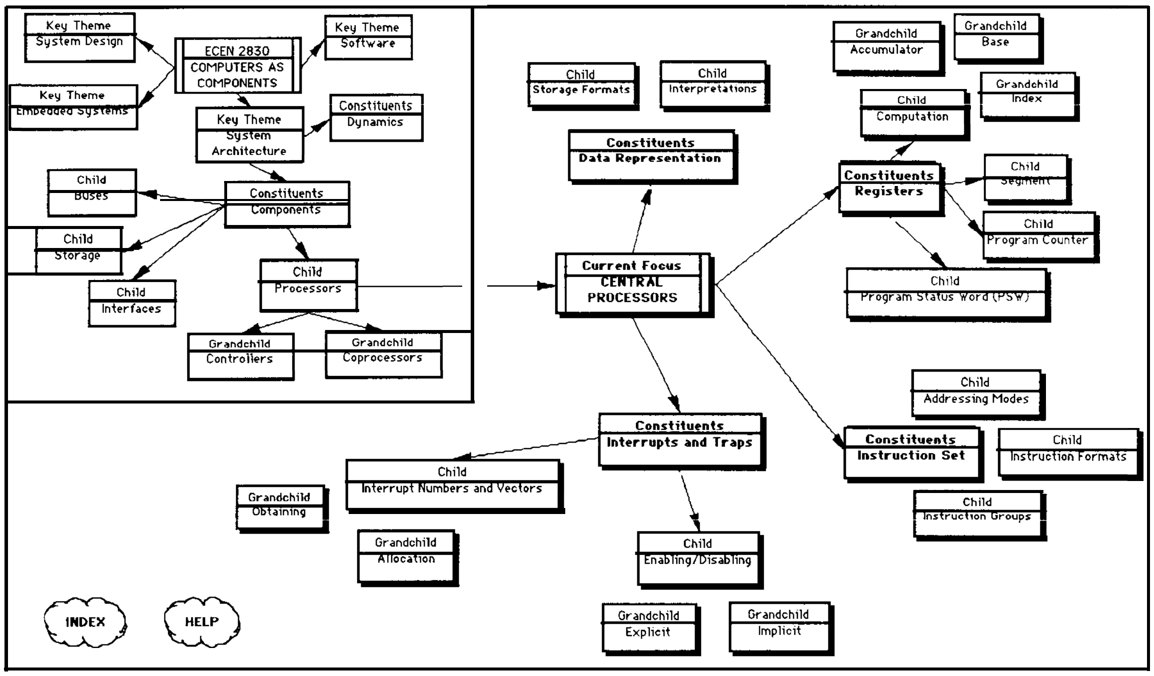
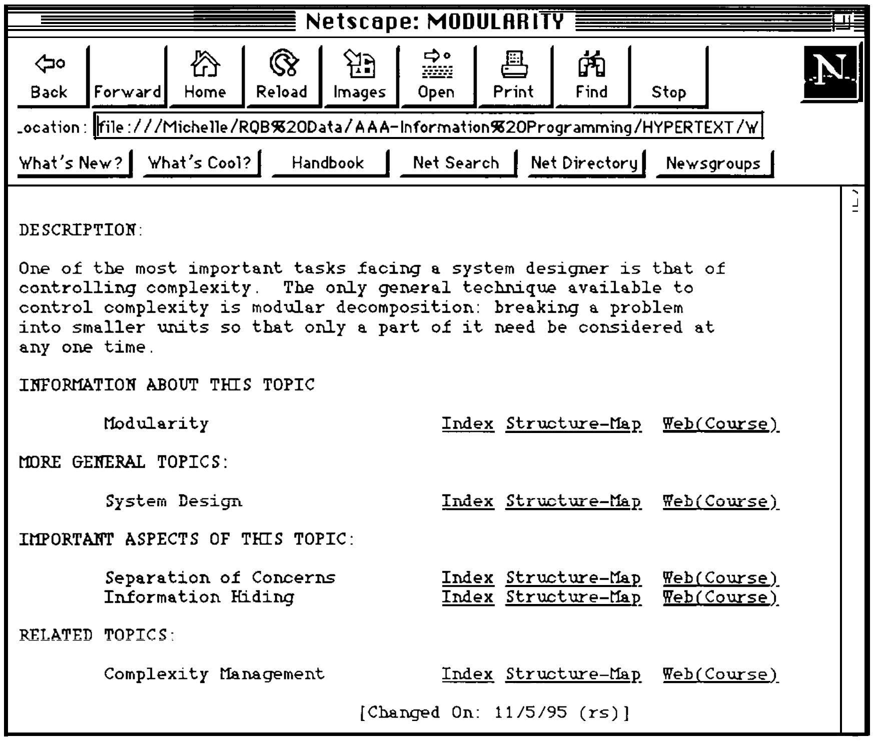


Figure 3
Lower-Level Structure Map

The structure maps provide an overview of the topic organization, allowing the user to maintain context at all levels. Link spaces, on the other hand, make available very rich contexts for the nodes to which they are attached. Because they are implemented as HTML files, they have the potential to contain graphics and animations. Conceptually, they are elaborations of the “See Also” pointers in a traditional index. (In fact, they were derived from that source. The index “See Also” trails and the course goals played against each other in developing the hypertext and link space structures.)

Figure 4 shows the link space associated with the node “Modularity” of Figure 2. It contains a descriptive paragraph about modularity and pointers (under “Information About This Topic”) to the textbook index, the structure maps, and the course syllabus. “More General Topics” is the parent in the structure map, and “Important Aspects Of This Topic” are the children. “Related Topics” are just that, related topics. They may be spatial neighbors in the structure map, but they need not be.



DESCRIPTION:

One of the most important tasks facing a system designer is that of controlling complexity. The only general technique available to control complexity is modular decomposition: breaking a problem into smaller units so that only a part of it need be considered at any one time.

INFORMATION ABOUT THIS TOPIC

```
        Modularity                 Index  Structure-Map  Web(Course)
```

MORE GENERAL TOPICS:

```
        System Design              Index  Structure-Map  Web(Course)
```

IMPORTANT ASPECTS OF THIS TOPIC:

```
        Separation of Concerns     Index  Structure-Map  Web(Course)
        Information Hiding         Index  Structure-Map  Web(Course)
```

RELATED TOPICS:

```
        Complexity Management      Index  Structure-Map  Web(Course)
```

[Changed On: 11/5/95 (rs)]

Figure 4
The Link Space for the "Modularity" Topic

The link space provides a much greater scope for elaborating the semantic context of a node than would be possible with the structure map alone. In effect, the link space is a textual semantic map in its own right. The structure of Figure 4 could be diagramed as shown in Figure 5. What the link spaces give us, in addition to descriptive power, is an additional dimension - the related topics dimension - that is not easily shown in the hierarchical structure map format. In fact, the dimensionality goes beyond the diagram of Figure 5 because for each node shown in that diagram, there might be links to the textbook index, to any of the structure maps, and to other material on the Web.

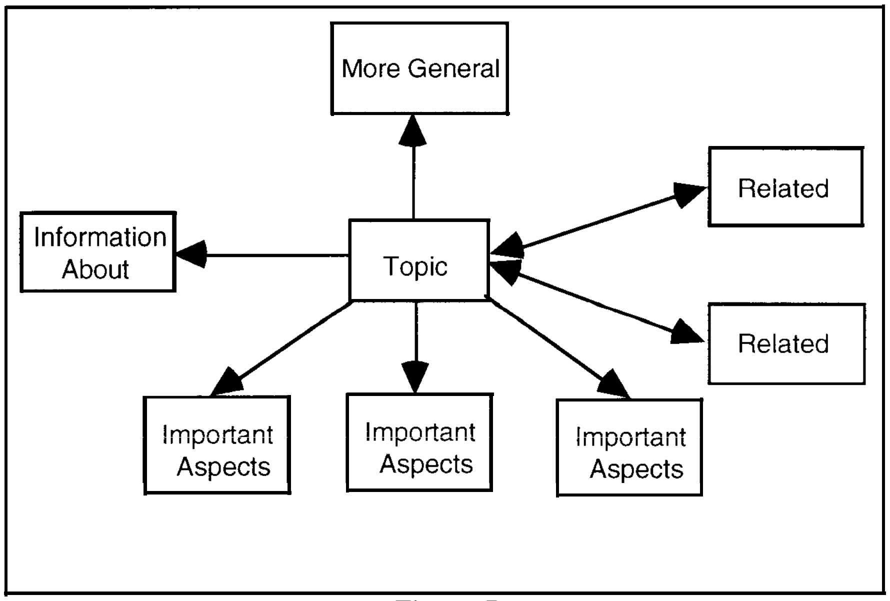


Figure 5
The Structure of a Link Space

Creating the link spaces was extremely labor intensive, far more so than we had envisioned when we started the project. In fact, it was so time consuming that actual deployment of the structure maps and their associated link spaces has been moved to the spring semester. The situation becomes even more critical when maintenance issues are taken into consideration. To handle this we have developed some important strategies for automating most of the maintenance tasks, permitting the focus to be on content development.

3.4. Maintenance

Maintenance is a serious problem, especially since this course is destined to evolve and web-reported research is changing rapidly. All components, including the syllabus, narrative, textbook analysis, and homework assignments, are undergoing continual change as experience with the course and new materials dictate. The link spaces, in particular, because of their number and volatility, present especially formidable update issues. In their current form, rearranging and relinking the basic descriptions is tantamount to a complete rewrite. In addition, we want to be able to edit the link spaces without the noise of HTML markup.

Story space provides a nice solution to these problems since it has both the ability to export HTML files and the ability to combine writing spaces in any desired order before exporting them. Thus it is a real hypertext editing environment that allows us to see and edit the structure as well as the text of our materials. It also has the ability to generate the one-to-many links that are crucial to the link space concept. Ironically, Story space is so much more powerful than HTML that we must be careful not to depend on facilities that have no analogs in the Web hypertext environment.

A more serious deficiency is that Story space has no versioning capabilities. This means that we will have to use a manual copy-and-rename procedure to handle updates, a tedious and error-prone process. Offsetting this is the crucial significance of the ability to make all changes in content and links to true hypertext files and then export automatically-generated HTML files as needed for Web installation.

4. Conclusions

It is a well-known fact that the lack of structure in web-based materials can absorb a great deal of time without commensurate benefits. We tried to avoid this problem by providing a very simple top-level organization where the major documents (news, syllabus, homeworks, grades and administrative details) were directly accessible. To encourage focused use of the resource material, we used a series of guided trails that connect the web-resident index, all of the web-based resource materials, and related research and courses offered at the University.

The guided trails are managed through the link spaces and the index. The link spaces, in their function as descriptive See Alsos, provide a rich set of choices at every point along the trail. The index provides an alphabetically organized and more detailed, though non-descriptive and less structured, approach to the guided trails. The index locators are either textbook pages or links.

Our students appeared somewhat baffled by the wealth of material at their disposal. Many would ask questions that indicated they had not explored even the major documents. We believe that it will be necessary to include homework problems requiring them to explore the concept net in order to get them started. (These assignments are similar to those that familiarize them with the computing environment.)

A formal evaluation of the class was carried out using a group interview: A team of two evaluators took over the class for one period. No prior announcement of the visit was made, and the professor and all TA's were excluded. The class was divided into small groups, and asked to come up with statements describing the strengths of the course and statements describing areas needing improvement. The only constraint was that each group had to unanimously agree on each of the statements they reported. After about half the period, the statements were voted on by the entire class. We show the results of that vote as Figure 6. Although there were 93 students registered for the course, only 51 were present that day. The attendance figure was typical, and may reflect a division of labor within study groups.

ECEN 2830 Evaluation, Fall 1995, W. Waite					
Strengths of Course	Agree	Disagree	% Agree	910 Disagree	No. Resp.
Like the hands-on experience in lab.	**50**	**0**	100%	o%	**50**
WWW material is helpful.	49	2	96%	4%	51
Prof. incorporates ideas not in text.	44	4	92%	8%	48
Labs build-well on previous labs.	45	5	90%	1070	50
WWW material is well-organized.	44	7	86%	14%	51
Group work is effective.	41	8	84%	16%	49
Office hours are enlightening.	39	8	83%	17%	47
Prof. is excellent lecturer.	42	9	82%	18%	51
TA's are helpful.	39	9	81%	1970	48
Professor and TA's very accessible.	37	10	79%	21%	47
Like the Monday homework assignments.	32	15	68%	32%	47
Labs relate well to lecture.	28	22	56%	44%	50
Exam reviews are helpful.	23	25	48%	**52%**	48
Areas Needing Improvement	Agree	Disagree	% Agree	% Disagree	No. Resp.
Need more computers.	50	1	98%	2%	51
Labs should get-more credit in grade.	49	2	96%	4%	51
Text not readable.	49	2	96%	4%	51
Need printer in lab.	49	2	96%	4%	51
Book needs more examples.	47	3	94%	6%	50
Better documentation of code.	46	5	90%	10%	51
Text written for those with experience.	46	5	90%	10%	51
Need more explanation of basics at beginning.	43	5	90%	10%	48
Serious problems with computers in lab.	44	7	86%	14%	51
Support code is cryptic (fewer global variables).	42	7	86%	14%	49
Should go over exams.	40	7	85%	15%	47
Should suggest relevant exercises.	40	10	80%	20%	50
More than 60 min. of material on exams.	37	10	79%	21%	47
Labs need more structure.	40	11	78%	229'0	51
Shouldn't be sudden increase of difficulty in assignments.	35	12	7470	26%	47
Should supplement Web pages with hard copy.	32	17	65%	35%	49
Should have assembly language course first.	31	18	63%	37%	49
Separate the lab period from the recitation.	31	19	62%	38%	50
Prof. should follow text.	16	35	31%	69%	51

Figure 6
Results of the Group Interview

The major complaint, a need for more computers, was interesting because the lab was underutilized most of the week. There was always a crunch just before the assignment was due, so the plea for more machines really was a request for us to support procrastination. The request for more lab credit indicates that the lab is regarded as a "product" rather than a study opportunity. We need to change that perception, because it is indicative of the fact that the students still don't feel really responsible for their own education.

There was general agreement as to the value of the hands-on experience in the lab, and students felt that the web material was helpful and well-organized. Group work was considered to be effective. Thus our major premises in the course design met with agreement from the students.

As a consequence, in the coming semester we will not modify the course goals but we will modify their implementation to take into consideration both the problems we observed and the problems students felt were impeding their ability to achieve the course goals.

First, to ensure that students use all the web-based materials, we will create homework assignments that depend on extensive use of the those materials. These assignments will involve the structure maps and index as well as the written materials. Second, we will aggressively address the issues of lab integrity maintenance through a combination of troubleshooting programs, modification prevention guards, and formal procedures. Finally, we will explore methods of using the web-based materials to guide student exploration of related courses and research available on the Web. You can view the current state of our project at http: //ece-www.colorado. edu/~ecen2830/.

References

[1]Bolter, Jay David and Michael Joyce. "Hypertext and Creative Writing," *Hypertext '87,* pp. 41-50. Baltimore: Association for Computing Machinery, 1987.

[2]Marshall, Catherine C., Frank M. Shipman III, and James H. Coombs."VIKI: Spatial Hypertext Supporting Emergent Structure," ECHT94., pp. 13-23. Baltimore: Association for Computing Machinery, 1994.

[3]Parunak, H. Van Dyke. "Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning," *Hypertext '91*, pp. 233-242. Baltimore: Association for Computing Machinery, 1991.

[4]Zellweger, Polle T., "Scripted Documents: A Hypermedia Path Mechanism," *Hypertext '89,* pp. 1-14. Baltimore: Association for Computing Machinery, 1989.

[5]Furnas, George W., "Generalized Fisheye Views," *CHI '86,* pp. 16-23. Baltimore: Association for Computing Machinery, 1986.

W. M. Waite, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309-0425, William. Waite@Colorado.edu

R. Simpson, Eastgate Systems, 134 Main Street, Watertown, MA 02172, skylark@eastgate.com