Introduction

One of the curriculum design problems facing computer science is the pressure to add new courses to cover additional topics. Often there is no curriculum space available. In some cases, the topics are covered more effectively by distributing them among a set of courses.

However, when parts of a topic, such as parallel processing or scientific visualization, are distributed among a set of courses, additional problems arise in areas of coordination and consistency of coverage.

A curriculum design paradigm is presented which seeks to formalize the distribution of a course in such a way that the integrity of the topics is maintained and documented.

Background

There have been two areas of computer science recently added to the curriculum at Taylor University: parallel processing and scientific visualization. In both cases, topics have been distributed across a large number of courses. In order to validate the new curriculum and to help ensure its consistency the Distributed Course model has been developed and applied. The two areas have been approached differently but have both benefited from this paradigm.

Over five years ago, the CS department became convinced that parallel processing should be an important part of the undergraduate curriculum. The curriculum was, of course, already too full. Therefore, it was decided to incorporate a variety of parallel processing topics in several courses.

Although this decision was originally viewed as a compromise, the distribution of parallel processing topics across the curriculum is now viewed as advantageous [1,2]. There have been many articles published debating the proper approach to introducing parallel processing in the undergraduate curriculum. Some argue that there are “many faces of concurrency” so that a separate course is needed to focus on the distinct “phenomenon” of parallel processing [3] while others state that it is not sufficient to add one or two courses to the curriculum but that integration across the curriculum is important [4] and that a radical change to existing courses is necessary [5].

The author has stated [2] his belief that parallel processing has matured to the point that it should be treated as a “spatial” topic rather than a “special” topic: that it should definitely occupy space in the CS curriculum but that it should not be viewed as a totally distinct sub-discipline. This view is based on the idea that one approach to viewing computer science as a whole is to approach the parts of the discipline as existing on a continuum from truly sequential to massively parallel. In this approach, many areas of computer science are viewed, to some degree or another, as parallel. Examples would be client/server models in a variety of courses, parallel paths in sequential hardware, and concurrency issues in operating systems.

If this view is held, distribution into many courses is an obvious approach.

An additional consideration is that parallel processing can be viewed as a tool to be used for purposes such as image processing and neural networks as well as an object of study by itself. In the curriculum at the author’s institution, many courses have been impacted by the parallel emphasis in the curriculum. This effort has been greatly aided by the receipt of an NSF grant (DUE-9552245 - Integrating Parallel Processing as a Tool Throughout the Undergraduate Computer Science Curriculum).

It is not as obvious, nor as widely done, to distribute visualization topics across several courses. Recently, the author and his department have decided that some topics from visualization should be included in the CS curriculum. Since a course can not be added, the natural idea is to investigate distribution. It is, perhaps, easier to distribute
visualization since a list of core topics is being developed [6] that can easily be organized into modules. The eight core topics are:

Theme 1: Introduction to Visualization
Theme 2: The Data
Theme 3: The User and the Tasks
Theme 4: The Mapping
Theme 5: The Representations
Theme 6: Interaction Issues
Theme 7: Concepts of the Visualization Process
Theme 8: Systems and Tools

However, the applicability may be viewed as more narrow. Once the decision is made to consider the distribution of visualization topics, some courses emerge as possibilities. For example, algorithm and data structure visualization can be used in a variety of courses in the CS curriculum. The 1996 SIGCSE Proceedings contains at least 6 papers discussing this area while the 1997 Proceedings contains at least 9 such papers. Other areas of visualization include mathematical, database and digital library, and business applications.

The two areas of computer science discussed above are quite different in terms of a desire to distribute across the curriculum and support for such distribution in terms of an evolving body of accepted topics as well as other areas. However, the same problems exist whenever a “course” is distributed in pieces over several actual courses.

One such problem is the need for coordination. If topics are inserted into existing courses taught by a variety of professors, the effect may be like buckshot fired at a flock of birds. Some of the topics are hit, but somewhat by accident, in that the flock is the target and individual birds are hit because of the number of objects moving in the correct general direction. A better analogy would include coordination somewhat like a missile with multiple, but independently targeted, warheads. Each piece in the distribution should aim at one part of the target material to be covered.

There are additional problems, some of which are addressed by the Distributed Course paradigm while others are not. These problems are discussed below.

**Paradigm**

A Distributed Course is a body of related material that is intentionally included in a collection of courses rather than one (or more) dedicated courses. It is possible that a topic distributed over a curriculum would include enough actual instructional time to be the equivalent of more than one course.

The process of developing a Distributed Course requires the following steps:

1. Identify the body of material to be included in the curriculum. If possible, determine the amount of time to be devoted to each.
2. Identify the courses where such material might be taught. If possible, determine the amount of time available.
3. Distribute the topics to the courses.
4. Document the Distributed Course in much the same way as a normal course.

Since the courses involved will be updated (and, perhaps, replaced) at different times and taught by different faculty, it is important that the choice of topics and distribution pattern be determined with as much input as possible. At our university, several of the department faculty were involved in the choice of topics and courses where the topics would be included.

It is important to note that the inclusion of a new topic in a course does not necessarily require a large amount of time. Many of the topics listed previously can be addressed with little course modification. For example, multiple instruction issue may be discussed in a computer organization course and the obvious point that this is a hardware parallel issue can be briefly stressed and related to other parallel topics. An example from visualization might include a discussion of an algorithm using a visualization tool and including a discussion of whether the visualization is effective and how it fits into the overall concept. Many graphics topics, especially in areas related to interface design or human-computer interaction, could be extended slightly to apply to visualization.

When the Distributed Course is developed with a formal approach as outlined above, there are some additional advantages.

Ownership. The members of the faculty have participated in the choice of topics and the mapping to courses they may teach. Thus, they have an interest in the initial and continued implementation and success of the Distributed Course.

Continuity. By feeling a sense of ownership, faculty are more likely to continue to include the material as the course evolves.

Commitment. The sense of department, as well as individual, ownership will encourage faculty who
begin to teach a given course to sense a departmental mandate to include certain material.

Coverage. If each faculty member considers how the general topic, such as parallel processing, fits into a course, there will be more areas and more relationships found.

Ease. It is not necessary to do all the paperwork required to add a new course through formal university channels. Therefore, a quicker response can be made to curricular needs that fit the paradigm.

There are some problems that arise from the fact that topics are distributed which are not solved by the Distributed Course approach. These problems, however, can be minimized by use of the paradigm. Obvious problems not solved are:

Lack of textbook. Most traditional courses use a text that addresses topics in its domain. There may be mention of the distributed topic, such as discussion of hardware parallelism in a computer organization/architecture text. However, there is not a text that ties together the concepts belonging to the Distributed Course. It should be possible to develop a local manual or text for the distributed course. Such reference would include the topics involved, the relationship between the topics and the traditional courses, relationships among the topics, and local information such as hardware and software that might be used in one or more traditional courses to support the parts of the Distributed Course. An example of local information would be PVM instructions that might be applicable to a variety of courses.

Lack of documentation for the Distributed Course. Documentation intended for the department and university can be embodied in a manual as described above. External documentation, such as appearance on a transcript, is more difficult. However, if a department maintains a set of web pages describing the curriculum and individual courses, a similar page could be developed for the Distributed Course. Prospective students, graduate school admissions officers, or employers of graduates could consult this form of documentation. There is still a large problem in that all students graduating from a department do not normally take exactly the same set of courses so that some students might complete all of the parts of a Distributed Course while others may not do so.

Examples

In our curriculum, the Parallel Processing Distributed Course has been in place for three years. Several regular courses have been modified. The curriculum has been represented as "parallel aware" in that many topics related to parallelism are indicated as such and included in the Distributed Course. The regular courses include:

1. Data Structures - first introduction and parallel sorts
2. Introduction to Artificial Intelligence - parallel applications for neural networks and image processing
3. Algorithm Design - bulk of more theoretical material
4. Data Communications - client/server model and communications needed for parallel processors
5. Software Engineering - Ada tasks and coordination
6. Database Concepts - distributed data bases, parallel database servers
7. Computer Graphics - parallel hardware and algorithms
8. Computer Vision - parallel image processing and pattern recognition
9. Natural Language Processing - parallel neural networks
10. Computer Organization - parallel nature of "sequential" processors, parallel architecture, quantitative measurement of communication times in parallel systems
11. Language Structures - parallel languages
12. Operating Systems - concurrency in standard operating systems and discussion of distributed OS
15. Information Systems Analysis - client/server approaches
16. Information Systems Design - client/server approaches

The department is currently in the process of devising a Visualization Distributed Course. Potential regular courses to be included are:

1. Data Structures - algorithm and data structure visualization
2. Data Communications - visualization of routing and traffic and communications considerations of distributed visualization
3. Database Concepts - digital library and related ideas
4. Computer Organization - visualization of processor and instruction activity
5. Algorithm Design - algorithm visualization as analysis tool
6. Computer Vision - discussions of human perception could be related to visualization (and interface design) issues
7. Computer Graphics - GIS and visualization modules included
8. Advanced Computer Graphics - more thorough discussion of visualization topics
9. Language Structures - visual languages such as Iris Explorer

It is undecided at present where more general topics such as elements of design will be included. The mapping of the eight core topics to the material covered in the courses is not complete.

There are additional CS topics that might fit the Distributed Course paradigm. Such topics are ones that quickly evolve and demand a place in the curriculum such as parallel processing and visualization.

Web programming is such a potential distributed course. There are many areas of web programming and interaction to consider: html, vrml, client/server, cgi, perl, java, database interface, CORBA, ODBC, etc.

Many of these topics could easily be included in existing courses. The approach taken by our department is a hybrid approach. A course, tentatively titled "Interactive Application Development," will be in our CS core and will cover some of the topics listed above. Other topics will be distributed. Currently html is covered in the general education CS course that is also required of CS majors and vrml is covered in Computer Graphics.

Implementation

Although one of the stated advantages of the paradigm is that formal university course approval procedures are avoided, there should be some documentation of the process. In our small department with seven CS faculty, the procedure has been quite informal and mainly verbal during department meetings.

There is a need for documentation after the topics are distributed. This documentation should take a form similar to that of regular courses. In our environment, a web page will be produced which lists topics much like an on-line syllabus except that the topics have links to other (regular) courses. There will be sections devoted to purpose and content of the Distributed Course.

The web pages for our Distributed Courses will be available from http://www.css.tayloru.edu/classes/DC. Figure 1 is a part of the parallel processing distributed course web page.

Conclusions

A curriculum design paradigm based on validating and documenting the distribution of related topics across many courses in a CS curriculum has been developed and is being used for two major curriculum areas: parallel processing and visualization.

While some of the tests of the success of the process, such as a continued commitment on the part of the entire faculty, require more time to assess, the procedure has worked well. More thought has been given to discover as many areas of interaction with the topic as possible and there is a sense of ownership of the distributed courses.

References


