Student-Developed Single-Concept Learning Modules

Charles R. Bauer, Computer Science Department, Illinois Institute of Technology, Chicago, Illinois 60616, bauerc@iit.edu

Abstract *— Surveys of college seniors will quite often reveal that, in their maj or courses, there were certain concepts that were difficult to master at the first encounter. By collecting this data, and then using students to develop web -based learning modules for the most frequently appearing problem concepts, first encounters to a difficult concept may be easier for students to master.*

Index Terms *— Single -concept, Web -based, Modules, Instruction*

INTRODUCTION

Peer tutoring has been found to be a very valuable asset to student learning. A recent survey of the teaching assistants/tutors for Computer Science courses revealed that certain concepts appeared often as problem areas. Student after student appeared at the teaching assistant's office stymied by the same lack of understanding of a basic concept. In the student-to-student tutoring/learning session, the teaching assistant possibly best understands the problems associated with a difficult concept. Quite often the teaching assistant had faced this same difficulty in the understanding of a concept when he or she first encountered it. At Illinois Institute of Technology peer tutoring has been used in many courses; in general, this technique has met with success. The biggest problem, limited hours of teaching assistant availability, is what we have tried to solve with the use of web-based single-concept learning modules to provide help with difficult concept understanding.

BACKGROUND

Most college/university professors consider their first teaching experience a disaster. They lecture for several weeks, then give a quiz or test and are shocked at the results. At Illinois Institute of Technology we offer two graduate courses that can make this first venture into the classroom a more rewarding experience for both the instructor and the student. Planning, developing and teaching a course requires three equally important components: 1) knowledge of the subject, 2) course content organization, and 3) presentation techniques. These three components have been determined by questioning many undergraduate and graduate students, and asking them to list single words which best describe the outstanding instructors they have encountered.

A sequence of two graduate courses is offered for Ph.D. students who are planning on a teaching career at the college or university level. The first (CS 560) in the sequence deals with organizational techniques for course planning, the second (CS 561) deals with the various presentation methods, from the whiteboard to multimedia approaches.

It was to CS 561 graduate students, in the spring of 1997, who were tutors and teaching assistants, that the idea was born to prepare for the web what would be called Single-Concept Learning Modules (SCLM). These are short tutorials or explanations that present information that would typically be given out in a person-to-person session. The topics of these Learning Modules would be obtained by interviewing undergraduates who had taken one or more required undergraduate Computer Science courses.

Each of the graduate students was given a list of required undergraduate courses and told to interview five to ten students who had taken each course. Each interviewee was asked to respond with two or three of the most difficult concepts encountered in each of the required courses completed. The concept was to be expressed in one or two words.

This information was gathered and summarized to find the most common problem concepts. Overall about 150 students were interviewed. The most common problem concepts were quite evident. The summary was then presented to the graduate students, and they, in turn, put the list in order of preference of concept they wished to develop. It worked out that each student received his or her first or second choice. There were no guidelines given as to length or style for the SCLM, the thought being that an unstructured approach would probably yield the best results.

The final version of their SCLM was due eight weeks into the semester. Preliminary versions were critiqued by all of the members of the class. The final versions were put up on a web page, and the information about the web page was then sent to all of the instructors of the courses involved. The instructors were asked to make the web page available to their students.

Each time CS 561 has been taught the same techniques have been followed. This has added to the list of SCLM's. Some of the SCLM's are better than others, but student reaction overall has been very positive. Comments and criticism have been requested from anyone using them. These SCLMs are to be considered a dynamic entity. Each semester, these first versions will be examined again and new versions will be explored and added to the web page. Many semesters will have to pass before any profound statements can be made, but first reactions seem to indicate that the idea is sound. Student comments have been favorable. A quote or two: "The greatest thing about this is that the web page is available 24 hours a day."; "Why aren't there more concepts on the page?"; and "When will other courses be covered?"

The web page address is:

http://www.cs.iit.edu/~cs561/cs_sclm.html

We would appreciate any comments, good, bad or just so-so. Some of the SCLM's are general in content and others very specific. Also some team-effort SCLM's have been added.

COURSE/SCLM

CS 105 Introduction to Computer Programming
- Problem Solving With Pseudocode 1
- Problem Solving With Pseudocode 2
- Iteration
- Arrays 1
- Arrays 2
- Functions 1
- Functions 2
- Functions 3
- File Streams (HTML from C++)
- Pointer 1
- Pointer 2

CS 330 Discrete Structures
- Discrete Probability
- Functions
- Trees
- Permutations
- Combinations
- Sets
- Boolean Algebra

CS 331 Data Structures and Algorithms
- Heaps
- Queue ADT
- Stack ADT
- Binary Search
- Sorted List
- Singly Linked List ADT
- Quicksort

CS 350 Computer Organization and Assembly Language
- Single Cycle Implementation
- Introduction to MIPS
- ALU Implementation
- ASCII Code

CS 351 Systems Programming
- Handling Input Devices
- Scroll Bar Messages
- Basics of Menus, Dialog Boxes or Resources
- Child Windows
- System Timer
- Network Programming
- Introduction to Network Programming

CS 425 Database Organization
- Integration of Structured Data and Text
- Relational Algebra
- Semi-Structured Databases

CS 450 Operating Systems
- Deadlock
- Disk Scheduling Algorithms
- Fork System Call
- Piping and Redirection
- Memory Management

CS 487 Software Engineering
- Dataflow
- Coupling
- Cohesion

Division Algorithms
Basics of Caching
Pipelining

CONCLUSION

We believe this SCLM approach could be used in any discipline and would be particularly successful in engineering courses where many new and sometimes difficult concepts are presented.

SAMPLE

What follows is a sample SCLM. Since these are made available as web pages much is lost when presented as hard copy.

DEADLOCK HANDLING

Deadlock *- A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.*

Necessary Conditions:

- unsharable resources
- hold while waiting
- circular wait
- no preemption

Unsharable Resources - resources that may be used exclusively
that is, with no other combination or sharing of resources.
e.g. magnetic tape, printer

Hold While Waiting - when process is allowed to hold resources
while requesting other resources.
e.g. process A holds R1 and R2, while it requests R3 and R4

Circular Wait - a state in which a cycle of processes exists such
that each process holds a resource that is being requested by the
next process in the cycle and that request has been refused.
e.g. process A holds R1 and R2, while it requests R3 and R4
which may be held by process B which is requesting R1

No Preemption - resources cannot be recovered from processes.
e.g. process A holds R1, and R1 cannot be recovered

Therefore, for deadlock to occur all of the four conditions must take place in the system.

Deadlock Prevention

To prevent deadlock incidents, at least one of the four necessary conditions should be removed.
Unsharable Resources Unsharable resources may be identified as printers, plotter, and magnetic tapes in which case cannot share resources. However, simple modifications to the handling nature of such devices will at least make these unsharable resources virtually sharable. A good example would be the implementation of a printer spooler. The spooler will print the output, which was stored in temporary files, as determined by the managing policy of the printing queue. Thus, a few processes may print simultaneously on the same virtual printer. **Hold While Waiting** There are two dominant manners in which the hold while waiting condition may be avoided. The first requires that each process will have to specify and acquire all resources needed at one time. If this is not possible, the process will wait until all necessary resources are available for use. The second states that the process will have to release its held resources before requesting another. Note that both of these methods significantly decrease resource utilization. **Circular Wait** The algorithm used as a remedy to the circular wait problem basically numbers resources and orders them into increasing sequence. Processes may have resource requests granted only if they hold resources with a lower number or priority than that of the last requested one. Therefore, a circle is never allowed to form. **No Preemption** To prevent no preemption, the OS must allow some form of preemption. Preemption of resources of a certain process will force the process to be rolled back to the point at which it acquired it.

Deadlock Detection and Recovery

This approach is based upon constant periodic system inspection for the deadlock state. Depending on system overhead, the approaches for detection will vary based upon time intervals of system inspection. Once detected, there are two possible recovery methods: process termination and resource preemption. Process termination or abortion is a very risky recovery method. Depending upon the resources that a process is using and when it is terminated, will drastically affect the system. In some cases, all processes may be aborted and the system starts off clean. Resource preemption - to terminate one process at a time calls for some type of victim selection. This may be based upon process or resource priority and time consumption. Keep in mind that starvation must be avoided when selecting termination victims. Never consider the ostrich algorithm in which case the system is supposed to pretend that the problem of deadlock does not exist. This approach does not seem very logical, but the reason for this is that the price for handling deadlocks may be extremely high with low probability of occurrence. Therefore, the deadlock problems are handed over to the system operator or the user.

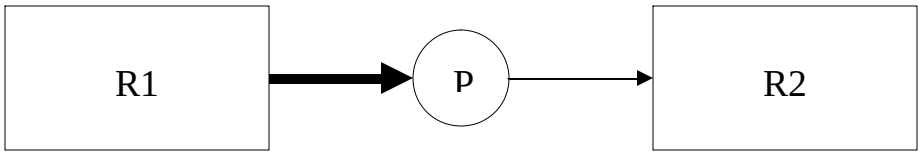
Deadlock Avoidance

Deadlock avoidance tries to avoid the deadlock state instead of coping with the prevention or recovery methods of deadlock. In such cases, the OS may pretend that the request is granted, and then run a deadlock protection procedure. So, if the simulator detects deadlock, then the process is not allowed to continue and is rolled back. However, this approach does not completely remove the possibility of deadlock. A better

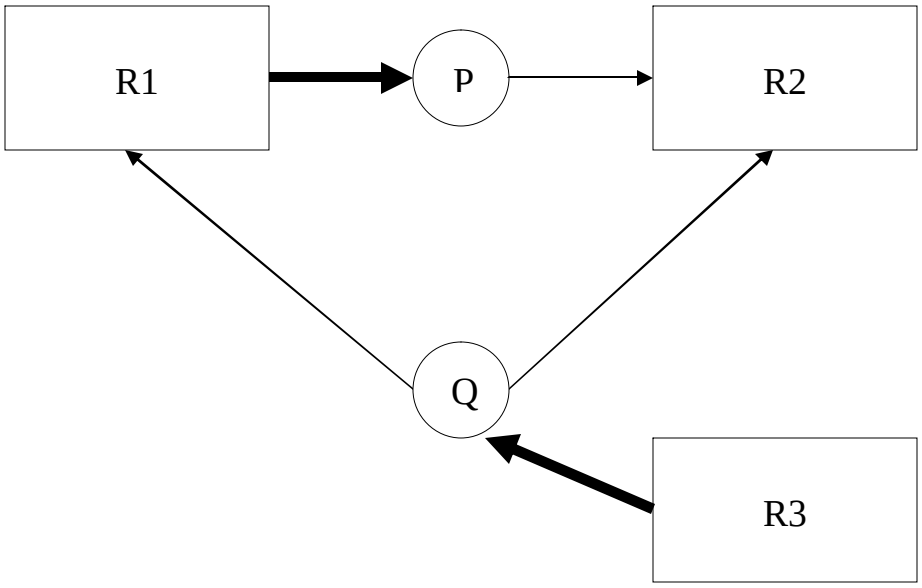
approach would be to make each process pronounce the maximum number of resources of each type that it may need. Then, it is dynamically investigated by a detection algorithm to guarantee it. However, these avoidance methods have a very high overhead, and this may outweigh its usefulness.

Resource Allocation Graphs

Rectangles represent different resource types and circles represent processes. Dark edges directed from resource to a process states that a process holds that resource. Light edge directed from a process to a resource states that a process requests a resource of that type. Process P holds the only instance of resource type R1, while requesting resource of type R2.



Process Q holds resource of type R3, while requesting resources of type R1 and R2. Process P holds the resource of type R1, while requesting resource of type R2.



REVIEW QUESTIONS AND ANSWERES

Problem 1) Are all of the necessary conditions for deadlock independent, or does one or more of them have to hold for another to hold? Explain your answer. 2) Is it possible to have a deadlock involving only one single process? Explain your answer. **Answer** 1) Hold and wait implies mutual exclusion; circular wait implies mutual exclusion, hold and wait and no preemption. 2) No. This follows directly from the hold and wait condition.