A Role-Based Adaptive CSCL Environment for Intensive Hands-on Teaching and Learning under Rigid Time Constraints

Horst F. Wedde, Frank Thorsten Breuer and Muddassar Farooq
Informatik III, University of Dortmund, 44221 Dortmund, Germany

{wedde, breuer, farooq}@ls3.cs.uni-dortmund.de

Key words: Learning Environments, E-learning, Didactics of Computing, Open Learning Systems, Role-based communication, Teaching methods

Abstract: Computer-supported Collaborative Learning (CSCL) is a widely accepted group-teaching and learning methodology. It focuses on communication and collaboration aspects among learning entities, between instructors and students. The contribution of this paper is twofold. Firstly we introduce a novel, role-based concept of collaborative learning where both instructors and students/student groups operate in combined teaching and learning activities while acting mainly autonomously. Secondly the beneficial learning effect of these methods (for students and instructors) is greatly enhanced by our new approach of reducing the course material to the essential underlying problems. In addition we shape the multimedia-based CSCL methodology according to the nature and character of the problem world of the course subject area. We apply our method to a classical area of computer operating systems (storage management) illustrated by the example of the famous 'Dining Philosophers' Problem'. The benefits of our approach are of particular use for senior-level undergraduate courses, for Life-Long Learning or for Distance Teaching and Learning programmes. In many of these areas intensive hands-on experience and competence has to be achieved under rigid time or learning environment constraints. In traditional classroom teaching these constraints would typically only allow for a superficial acquaintance with the subject areas.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35663-1_34

1. INTRODUCTION

During the past few years, European universities have redefined their education programmes by introducing B.Sc. and M.Sc. degrees. In particular the strong and increasing demand for computer experts in industry has accelerated this development in Computer Science. However, a key problem has to be resolved at the senior level of undergraduate education. After acquiring programming skills and knowledge about program structures, algorithms and application areas that are immediately needed in professional practice, students are supposed to acquire an overview of research and development in core areas of Computer Science. At the same time they are expected to develop the basis for expert understanding needed in their professional lives. This should enable students to catch up independently with the rapid changes in Computer Science. In the traditional European curricula these objectives are achieved through graduate coursework as there is insufficient time during the period of undergraduate education to cover all the topics. It has been difficult, often unsatisfactory, both for instructors and authors of textbooks, to follow the traditional format of teaching because the presentation, albeit paramount, has frequently to be superficial.

We propose the novel approach of condensing the contents of a subject area and filtering out the essential substance and methods of their treatment as defined by a subject expert. We model teaching and learning strategies according to the characteristics of the essential elements (processes and methods) thus combining the reduction effect with a customised teaching and learning method. This results in a deep, hands-on experience in the subject area. These methods apply to all subject areas in distributed computing. They are equally relevant for revising multimedia and CSCL methodologies in Life-Long Learning programmes as well as in Distance Teaching (Bourguin and Derycke 2001; Koschmann 1996; Restsa et al 1999; Papert 1993; Brereton et al 1998; Gifford and Enyedy 1999). In these areas, most programmes and courses have to adhere to:

- a rigid compaction of the material and its treatment;
- multimedia based CSCL methodologies.

2. CASE STUDY

In a course on operation systems a key subject area is storage management. Processes are regularly stored on hard disks yet have to be resident in main memory for execution. Three major system functions interact to support the execution: *Main memory management, disk management,* and *long-term scheduler* (Silberschatz et al 2000). The first two attempt to manage their space *as efficiently as possible, to accommodate as many processes as possible,* at the shortest possible notice. The latter goal allows for a *high amount of concurrent execution.* The long-term scheduler determines which executable part of a process should next be loaded to main memory for execution. Such executable sub-processes (and necessary data) may be swapped into main memory or back onto disk. This only becomes known at execution time. Since the computation may change the volume of process data, the available space in main memory or the required space on disk, are dynamic parameters for management functions.

The long-term scheduler learns about (dynamic) priorities only when it is scheduling processes on disk to be transferred to main memory. The same is true for the speeds and priorities. In turn, execution times for the three system functions are not predictable - nor are their speeds and space requirements. Thus, although these *system functions* are sequential subprograms under the (centralised) control of a sequential monitor process they *appear to co-operate independently and autonomously.* The co-operation comes about as they execute their algorithms in an interleaving fashion, through the invocation by the monitor. The same is true if one abstracts from the service layer and considers user processes that share resources.

The unpredictability of system or user process behaviour means that:

The essence of storage management is to design the corresponding service functions as autonomous processes that collaborate in a framework where no assumptions on (relative) speeds, time of occurrence, priorities or centralised control could be utilised.

The problems involved are the key problems behind efficient storage management and are very sophisticated. If their nature is fully understood it is easy to perceive both the motivation and the interactional framework of the algorithms involved. If one tries to condense (traditional) storage management to its essence then the diverse algorithms can be dropped as a detail of secondary relevance.

Therefore it is very important to formulate a novel CSCL approach based on matching the subject structure with a novel teaching and learning role behaviour. Concentrating on the essentials of the subject will give the students the ability to deduce the importance of the key concepts, a desirable ability in any research and development environment.

2.1 The Dining Philosophers' Problem

The Dining Philosophers' Problem (Winowski 1981) is a vivid example of this class of problems. Consider five philosophers, who spend their lives thinking and eating, sitting at a round table with a bowl of spaghetti in the middle. The table is laid with five forks, one between each neighbouring philosopher (Figure 1). When a philosopher eats, he needs his left and right hand forks - hence he prevents his neighbours from eating. When the philosophers think they do not eat and therefore do not need their forks.

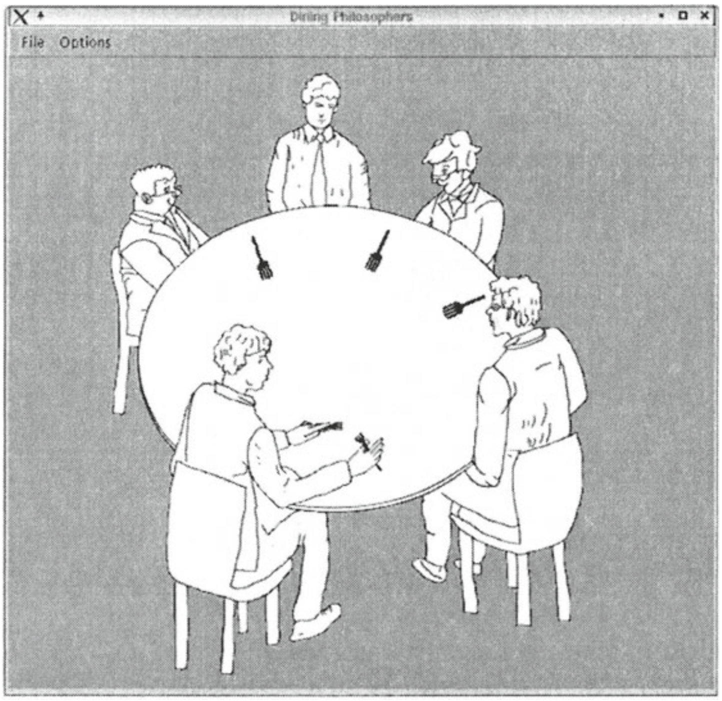

Figure 1. The Dining Philosophers

This is a simple representation of a finite set of concurrent, autonomous processes (philosophers), which run mainly independently (thinking). Occasionally they perform an action (eating), called critical section, during which they need exclusive access to a subset (their two forks) of the resources (all forks).

The solutions must meet the following conditions:

- *mutual exclusive access to the shared resources* (each fork can be used by only one philosopher at the same time);
- *no assumptions on the speed, the execution-time, and the need for resources* (the philosophers represent autonomous processes);
- *symmetry* (all philosophers have equal rights);
- *fairness* (no starvation);

- *no deadlocks* (the solution must not create a situation in which the philosophers block each other so that one or all of them can move any more towards resource access);
- *efficiency* (minimum overhead and waiting-time).

A simple solution is as follows. A philosopher who wants to eat, takes his left hand fork, if available, otherwise he waits for it. Then he tries to pick up his right hand fork in the same way. This solution fails when all philosophers get hungry simultaneously. Each grabs his left hand fork and will wait for his right hand fork endlessly, because nobody will give back his left hand fork. From this moment on the group is stuck - a situation called deadlock. Deadlock situations may be prevented if the philosophers give back the first fork when the second one is not available, or the philosophers take the first fork only if they are sure that the second fork is available as well.

The students should experience how co-operation must be organised so that all conditions are met. They shape solutions by assuming the role of philosophers. In a classroom environment the instructor mediates the discussion. Only a few students can play the role of the philosophers. The others perform the student role discussing the solutions. A CSCL environment allows all students to interactively perform the roles in this scenario and act like collaborating autonomous processes.

Once a deadlock-free solution has been developed students are encouraged to explore other aspects like starvation-freeness. This CSCL approach that realises interaction among students and instructor is greatly superior to traditional classroom teaching.

3. ROLE-BASED ADAPTIVE LEARNING ENVIRONMENT

Our novel approach is centred on a role-based communication model formulated out of Content-Centred Activity. We assign simple and compound roles to all those involved in the teaching and learning environment. All participants are autonomous processes, there is no central command unit that co-ordinates the syntax of the communication pattern among different actors. While playing a role, a number of bi-directional communication channels are established between the actors on which information, knowledge and experiences are exchanged. These communication channels play an important role in understanding the scenario and the context relating to a concept. We therefore propose a framework that provides an abstract modelling of the syntax of roles and communication channels and then performs content-specific refinements to add the semantics to the syntactic model.

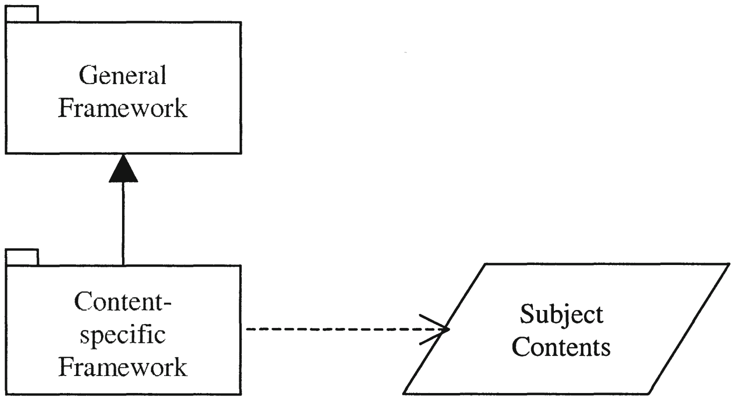

Figure 2 The Role-based framework

The idea of a role-based framework is to model instructional processes in a university lecture environment to minimise the misunderstanding of concepts presented in a lecture. We have two models - the general framework which abstracts the syntax of roles and communication channels in a learning environment, and the content-specific framework which provides the semantics to the roles and communication channels depending on the contents of the subject to be taught (Figure 2).

3.1 General Framework - Roles

The general framework realizes three roles in a learning environment

An Educator is responsible for imparting the concepts related to any subject area. He organises the contents according to a simple principle - "To present the unknown concepts in the form of known concepts".

A Broker is responsible for reducing the burden on, and the amount of queries to, the educator. Students should first contact the broker with a request for information they need. A broker, by virtue of his knowledge and experience, tries to reply to their requests. Should he not have the information in his knowledge base he contacts the educator for help. In a learning environment many brokers can exist.

The Student is at the focus of a learning environment. A large number of students exist in a learning environment and they form learning communities or groups to get a better understanding of the information they receive on a communication channel from the educator and brokers. Communication between students is the most important factor that helps in understanding a concept. The students are asked to assume the roles of different concepts and then simulate the context of the environment to propose solutions to the problem scenario. After performing the roles, they can contact their broker and receive feedback on the simulation.

3.2 General Framework - Communication Channels

The general framework emphasises four communication channels in the learning environment (Figure 3).

Educator-Student: On this communication channel the information, knowledge and experiences of the educator flow to the students.

Broker-Student: The Educator-Student channel is time bounded while the Broker-Student channel is time relaxed. Students can simulate the concepts in a more involved and deeper way. A broker provides feedback to a smaller community of students.

Educator-Broker: A Broker creates a pedagogical script (by discussing with the educator) for the students so that they perform their roles in an effective and informative manner. He provides feedback about the broker-student channel to the educator.

Student-Student: Ideas, algorithms, and discussion flow on these channels as students perform the roles assigned to them by the educator and brokers.

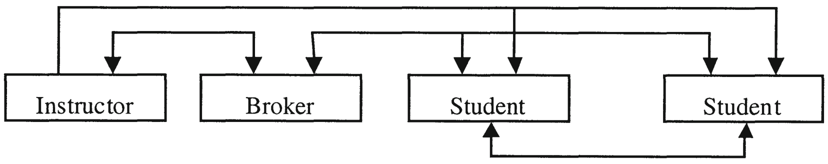

Figure 3. Communication channels in the learning environment

The efficiency of the communication channels is measured by the quantity and quality of information, knowledge and experiences that are exchanged. The quantity depends on the role a participant is playing. However the quality of a communication channel depends on the abilities of the person performing the role, the scenario in which the role is being performed and the number of people involved.

3.3 Content-specific framework

The co-operation among Autonomous Processes, as discussed in the case study, is the essence of teaching operating systems. It is reasonable to consider the students, brokers and educators as autonomous, collaborating processes when teaching operating systems.

Kernel/Educator

In this framework we consider the educator giving a lecture to act as the kernel. The students play the role of user processes. The kernel is the most important process on a computer (Silberschatz et al 2000) since the other processes depend on it for scheduling and execution. The kernel provides the interfaces for the co-operation of user processes with the kernel and among each other. During a lecture the educator plans, co-ordinates and manages the flow of presentation and discussions and allows questions.

Virtual Machine/Broker

Tutorials offer another classroom environment. Brokers present solutions to homework assignments and lead discussions on unsolved problems. We map the brokers to kernels on virtual machines. Virtual machines create an illusion that a process has its own processor and kernel. They enable processes to perform actions without directly interacting with the real kernel.

Scheduling/Discussion

In modern computers the CPU executes multiple processes concurrently by switching among them. CPU scheduling deals with the problem of deciding which process is to be run. Mapping the CPU-scheduling to chairing a discussion, where only one person may speak at a time, illustrates a problem of the subject area.

Discussions in group-learning environments are a representation of process synchronisation and resource scheduling problems in operating systems.

Context Switch/Questions

Switching the CPU to another process requires a context switch. The kernel saves the state of the old process and loads the saved state of the scheduled process. Although necessary, context-switch time is pure overhead - the system does no useful work while switching.

A comparable situation is a student asking a question during a lecture. It takes time for others to become aware of the student's inner communication context, which is necessary to understand the question entirely. Modern operating systems solve this problem by introducing threads. Processes can contain multiple threads of control, which share most of their information. Communication and context-switches between threads of the same process have less overhead due to the mutual context. We consider the working groups as processes and the students as threads.

Threads/Students

The students play the role of collaborating threads. In a classroom environment with a large number of students the communication overhead can exceed a tolerable limit. The students form teams and act like the threads of a process. They are familiar with each other's knowledge obtained by learning together. If the group wants to take part in a discussion the speaker addresses the other students or the educator. This concept makes an efficient discussion with a large number of students possible. The educator has only to pay attention to the speakers.

Communication

Collaborating processes can communicate using shared-memory or message passing. Communication by shared-memory means one process writes data into a shared buffer, another process reads from the buffer. Students perform similar actions when they collaborate on homework assignments. They read from and write on a piece of paper. Message passing is a form of inter-process communication very similar to human conversation. Processes send messages to each other. In both cases the students learn about the mechanism that allows processes to communicate and synchronise their actions by acting like collaborating processes.

4. CONCLUSION

We have outlined a novel principle of teaching and learning under heavy time or communication constraints such as in Distance Teaching or Life-Long Learning programmes. Our approach is based on matching the structure and behaviour in the teaching and learning environment with the structure and behaviour of the subject matter processes. We have explained in our example of an undergraduate-level course on operating systems how we distill the true nature of the problems and processes out of the traditional teaching material. This condenses the presentation and provides an adequate, hands-on experience and understanding of the essence of operating systems. Well-known CSCL methods have been combined with a novel role-based model of the teaching and learning environment. We have improved the instructional productivity and the learning quality for both students and instructors.

We are working on a completely interactive teaching and learning environment based on individual laptops with wireless connections. This will be the subject of forthcoming publications.

REFERENCES

Bourguin, G. and Derycke, A. (2001) Integrating the CSCL Activities into Virtual Campuses: Foundations on a new Infrastructure for Distributed Collective Activities. In *Proceeding of Euro-CSCL 2001.*

Brereton, P., Lees, S., Gumbley, M., Boldyreff, C., Drummond, S., Layzell, P., Macaulay, L. and Young, R. (1998) Distributed group working in software engineering education. In *Information and Software Technology*, No. 40. pp. 221-227.

Gifford, B. R. and Enyedy, N. D. (1999) Activity Centered Design: Towards a Theoretical Framework for CSCL. In *Proceedings of Computer Support for Collaborative Learning.* pp. 189-197.

Koschmann, T. (1996) Paradigm Shifts and Instructional Technology: An Introduction. In *CSCL: Theory and Practice of an Emerging Paradigm*, T. Koschmann (ed.), Lowrance Erlbaum Associates.

Papert, S. (1993) *The Children's Machine: Rethinking School in the Age of the Computer.* Basic Books, New York.

Resta, P., Christal, M., Ferneding, K. and Kennedy, A. (1999) CSCL as a Catalyst for Changing Teacher Practice. In *Proceedings of Computer Support for Collaborative Learning*. pp. 488-495.

Silberschatz, A., Galvin, P. and Gagne, G. (2000) *Applied Operating Systems.* John Wiley & Sons, Inc., 1st edition.

Winkowski, J. (1981) Protocols of accessing overlapping sets of resources. In the *Information Processing Letters*, Vol. 12, No. 5. pp. 239-243.