Critical concepts in the development of courseware for CS closed laboratories
Janet Mei-Chuen Lin, Cheng-Chih Wu and Guey-Fa Chiou
Department of Information and Computer Education
National Taiwan Normal University
No.162, Ho-Ping East Road, Sec.1
Taipei, Taiwan 106
{mjlin, chihwu, gueyfa}@ice.ntnu.edu.tw

Abstract
In this paper we present some of the experiences that we have gained from having developed a number of laboratory-based courseware for teaching computing concepts. We point out that lab-based teaching is a special category of computer-based instruction (CBI). The intended lab usage imposes certain constraints on lab-based courseware such that the methodologies for developing traditional CBI software do not seem to fit the development of lab-based software very well. Lab-based software requires special considerations as to the presentation of its course content and the design of its user interface.

1 Introduction
Ever since the joint ACM-IEEE Curriculum Task Force recommended that introductory computer science courses be supported by extensive laboratory work [4,10], an increasing number of researchers have been engaged in investigating the diverse aspects of incorporating closed laboratories into teaching computing concepts, and many lab-based software systems have been developed by different institutions worldwide. We have noticed that most of the research results reported in this field are about what the software systems are, how they are used in the lab settings, and/or their effectiveness in teaching and learning. Seldom did we find discussions about how a courseware specifically tailored for lab use should be designed, what the general guiding principles are, and what procedure should be followed to build a pedagogically sound laboratory courseware.

The authors of this paper have been engaged in the research into lab teaching in computer science since early 1994. Our research projects have been funded by the National Science Council of Taiwan and are scheduled to last till July 1997. Our ultimate goal is to establish an instructional model for incorporating closed laboratories into the computer science curriculum. In the past two years we have selected a number of topics typically taught in the CS1-2 courses for developing lab-based courseware.

The software systems that we have developed include SimCPU which simulates the execution process of a program by the CPU (Figure 1) [12], NUMBER which is about the internal representations of numeric values as well as conversions between different number systems (Figure 2), SimLOGIC for teaching combinatorial logic (Figure 3), SimSORT which animates sorting algorithms (Figure 4), SimRECUR for teaching the recursion concept (Figure 5), and SimLIST which visualizes the operations of the linked list data structure (Figure 6) [12]. A fuller-scale system which encompasses the major constructs of PASCAL is currently under development.

We must admit that all of these software systems have been designed and developed in an ad hoc manner in the sense that there were no well thought-out design philosophies behind them. We simply followed our intuition when any design decisions had to be made. However, as our software systems were taking their shape one after another, our experience accumulated and we began to think more deeply into the issues surrounding the design of lab-based software. We found that some of these systems did not turn out to be appropriate for lab use though they might be considered good CBI products. What are the characteristics of lab-based software? Are the fundamental principles and the methodologies, such as the ones we may find in [1,5,8], for designing effective CBI software also applicable to designing lab-based software? If not, how should they be modified to take into account the characteristics of lab-based courseware? The ideas presented in this paper resulted from our attempts to answer the above questions.

It is worth pointing out here that our following discussion focuses on the reinforcement labs only, that is, the lab activities that are intended to reinforce students' understanding of certain concepts already covered in the classroom lectures. There are, of course, other types of lab activities, such as the comparison labs, the improvement labs, and the discovery labs, as described by Naps [7]. Many more types of laboratories can be identified from traditional laboratory science courses. Notable among them are deductive laboratories, technical skill laboratories, problem-solving laboratories, etc. [3]. Apparently some types of laboratories will be much more difficult to design than others. For example, implementation of laboratories with higher levels of openness [9] would be much more complicated. Since we are currently more experienced in the reinforcement type of laboratories, and further because of our belief that the reinforcement lab is the more popular type of labs, our opinions given below are based on the reinforcement labs only. However, it should also serve as a foundation for discussing other types of laboratories.
The remainder of this paper is organized into four sections. In Section 2 we discuss the constraints imposed on courseware by its intended lab usage. Section 3 is an investigation into the instructional factors relevant to the design of lab-based courseware. In Section 4 we propose several guidelines for developing lab-based courseware. Section 5 consists of concluding remarks and a description of our future research directions.

2 The constraints on closed-lab courseware

What distinguishes closed laboratories from open laboratories is schedule, structure, supervision, and supplement. Schedule means that students complete a closed lab by attending a scheduled session, usually 2-3 hours long, at a specific facility. [10] Structure refers to the well-planned activities for the lab sessions, and supervision means that students' lab activities are carried out under supervision of the instructor or the lab assistant. Supplement refers to the fact that closed laboratories are typically used to supplement lectures instead of standing alone. Hence these four S's help determine the characteristics of lab-based courseware is elaborated in the following subsections.

2.1 The time constraint

With a general CBI software that is not intended for use in closed labs, it is seldom the concern of the software designer how much time the learner will or should spend on the software in order to achieve the desired learning effects. The CBI software designer is usually free to put as much material as is deemed appropriate into a piece of software, or to present the information in whichever way is chosen, as long as the memory space it takes up does not exceed hardware capacities. Designers of lab-based courseware have no such freedom. They should always keep in mind that the courseware as a whole, or an independent module of the courseware, is to support lab activities that are to be completed by students in two to three hours. With limited amount of time available for a lab session, students cannot afford to be distracted by superfluous information presented by the courseware. As a result, lab-based courseware tends to have a narrower focus of its subject content. It typically covers the key concept(s) of a certain topic rather than the complete material of the topic. The time constraint also demands that the user-interface for lab-based courseware be designed with special care. Students' learning activities in the laboratory should not be hindered by a user interface that requires much time to learn or is awkward to use.

There is another type of constraint to be considered. In addition to the fact that the time available for each lab session is usually two to three hours long, there is a limitation on the total instruction hours available for a course in a semester. The closed laboratories usually have to compete with lectures for those limited instruction hours. They should, therefore, be scheduled only when the particular lab experience is very necessary. The implication here is that not all topics of the course content are suitable for developing lab-based courseware. Closed labs should be used mainly to support lectures on those topics that are found most difficult to understand by students. This and other criteria for selection of suitable topics for lab-based courseware will be discussed later.

2.2 The structure of laboratories

To ensure that students achieve the best learning effects within the limited lab hours, lab activities should always be carefully planned. A step-by-step procedure is needed to guide students' learning process through the lab session. With closed-lab teaching, the step-by-step guidance may be provided by the instructor, the CBI software, the supplementary lab manuals, or any combination of the three. It is impractical for the instructor or the lab assistant to lead the lab activities because the progress will be interrupted continually by questions from students or the necessity to perform troubleshooting on hardware or software failures. The second alternative, i.e., building the control sequence into the lab-based software, is normal practice in traditional CBI software design. However, it is not a good choice for the design of lab-based software due to the following two reasons.

(a) Lab-based courseware for teaching computing concepts is most often used to simulate or visualize the internal process of abstract concepts. It is rather unnatural to interweave control information with the visual views presented on the computer screen. For example, our SimCPU courseware (Figure 1), which simulates the execution process of a program by the CPU, contains only one screen of material. Everything to be seen or done is on this screen. If we wish to build the control information into this courseware, one easy way is to include an extra button called, say, 'next step,' for learner to request for the next instruction, and use a pop-up window on top of the screen to display next instruction to the learner. It is feasible technically, but it will unnecessarily destroy the integrity of information on the screen.

(b) Provision of flexible learning paths in a CBI program has been a major issue studied by the researchers of the ITS (Intelligent Tutoring Systems) field. It is considered a very complicated issue, and a system that exhibits satisfactory degree of flexibility will be very difficult to implement.

On the basis of the above explanations, we believe that the use of lab manuals to specify the procedure of lab activities is the most natural and cost-effective way to structure a lab session. It is no wonder that seldom did we see a lab-based courseware without being accompanied by lab manuals. Worksheets are usually provided as part of the lab manuals for students to record the data collected or phenomena observed in the lab experiment. In order for a worksheet to serve its important function of a thought-provoking aid, it should contain well-designed questions that encourage students to think about what they see instead of filling the sheets out mindlessly.

2.3 Supervision

As mentioned previously, closed-lab activities are supervised closely by the instructor or a qualified lab assistant. Moreover, the courseware is often used together with lab manuals. With the presence of a supervisor and the lab manual, the students can ask for help any time difficulties are encountered in using the courseware, or the lab manual can be read for detailed information about the lab activities. For this reason we consider provision of extensive on-line help as one of the less important features in designing courseware for closed labs. However, we have found that sometimes on-line help can be used for quite a different purpose in lab-based courseware. Since a lab-based courseware is seldom a tutorial CBI, as will be explained later, it usually does not present complete information about a topic. The on-line help feature can be used to provide a brief synopsis of information related to certain knowledge items on the screen. For example, in our SimCPU courseware (Figure 1) we have implemented the on-line help feature with a pull-down menu that provides such
information about the syntactic rules for writing statements in assembly language, a sample program in assembly language, and so on.

2.4 Pre-exposure to subject contents
Laboratory work can be closely coupled with what is occurring in the lecture, or it can stand on its own, not in support of lecture work. When there is close coupling between laboratories and the lectures, lab-based courseware is not expected to carry the load of total instruction on a certain topic. It is rather to enhance students' understanding of those concepts that they have difficulties grasping by listening to the lectures only. Under such circumstances, the designer of lab-based courseware can assume that the concepts to be presented in the courseware are not completely new to the students, and that they are already familiar to some degree with the contents covered by the courseware. This explains why we mentioned before that a lab-based courseware is seldom a tutorial CBI.

3 Instructional factors relevant to the design of closed-lab courseware
In Section 2 we discussed how the nature of closed laboratories has resulted in a number of characteristics which differentiate a lab-based courseware from a general CBI program. In this section we investigate the similar issue, but the treatment is at a level that is more pedagogical than technical.

3.1 The phases of instruction
The process of instruction, either the classroom instruction or CBI, can be roughly divided into four phases, that is, presenting information, guiding students' initial interaction with the material, practicing the material by students, and performance assessment [1]. Considering that laboratories are used mainly as supplement to lectures and that the amount of time available for a lab session is rather short, it is both impossible and unnecessary for all the four phases to occur in a CBI courseware. We would expect a lab-based software to be capable of hitting right on the target, meaning that the courseware will concentrate on the practice phase only. To put it in another way, phases one and two should be considered the warm-up process that prepares students for the lab, and they should have occurred in the classroom instruction or the pre-lab tutorial. As to phase 4, i.e., assessing students' performance, it can be done in the traditional way, i.e., not on the computer, to avoid taking up extra lab time.

It is worth noting that, depending on the topic covered, practicing of material in a lab-based courseware may be very different from the computerized drills typically found in general CBI software. For example, our NUMBER courseware (Figure 2) deals with internal representation of integer and real numbers and also the binary and hexadecimal number systems. There are few difficult concepts related to this topic, and the purpose of the courseware is rather to enhance students' fluency in conversion between different number systems and the arithmetic operations of binary numbers. As a result, the practice provided by NUMBER is similar to the drill-type CBI involving mathematics. However, for lab-based courseware that is to enforce students' understanding of difficult concepts, practicing is more of the select and observe type. For example, with our SimSORT courseware (Figure 4) which visualizes the process of three sorting algorithms and SimRECUR courseware (Figure 5) for teaching the concept of recursion, the students are guided by well-planned procedures to press certain buttons or modify certain parameters on the screen and observe changes on data or phenomena.

3.2 The instructional events
Gagne's nine events of instruction [11] are often used as a logical framework for discussing general instructional design techniques and teaching strategies for CBI. The nine events are (a) gaining attention, (b) informing the learner of the lesson objective(s), (c) stimulating recall of prior learning, (d) presenting stimuli with distinctive features, (e) providing learning guidance, (f) eliciting performance, (g) providing feedback, (h) assessing performance, and (i) enhancing retention and learning transfer. As explained previously, students are supposed to have been introduced to the content knowledge covered by the lab software, and the software is used to reinforce students' understanding about some specific concepts within the topic. Under such circumstances, not all of the nine events seem relevant to the design of lab-based software. In other words, we do not think that lab-based software needs to instruct in the ordinary sense of instruction as a CBI software needs to. As stated in [11] that 'there are, of course, conditions in the learners and in the learning situation that modify the particular ways in which these events are delivered.' It is obvious that closed laboratory represents one of such situations. But how should we modify the way in delivering these nine events?

Students attending a closed lab session are usually given a pre-lab tutorial on the lab activities by the instructor or the lab assistant, and we may thus assume that their attention has already been gained through this pre-lab tutorial. It is also reasonable to assume that the learners have already been informed of the objective of the lab in the pre-lab tutorial. Therefore, events (a) and (b) can be excluded justifiably from the concerns of a lab-based courseware designer. So is event (c) since stimulating recall of prior learning is more important for new learning, which is not the case in learning with a lab-based courseware. The structured and the supervised nature of closed laboratories ensures that learning guidance is provided. As such, event (c) can be omitted too. The feedback given by a lab-based courseware is very different from what we would expect of a tutorial or a drill-and-practice CBI program. It is subtler and often in terms of changing data in some part of the computer screen. For example, if you delete a node of a linked list improperly when you interact with the SimLIST courseware (Figure 5), the system will respond by adding some more nodes to the dangling node area instead of telling you that you have done something wrong. As to events (f) and (h) which refer to assessing students' performance, they do not have to be provided by lab-based courseware because the tests had better be done in the traditional way to conserve lab time, as we have mentioned before. We are thus left with event (d), presenting stimuli with distinctive features, and event (i), enhancing retention and learning transfer. All the other instructional events are provided by the instructor. We can, therefore, conclude that lab-based courseware is, in fact, a most simplified form of CBI programs.

3.3 The issue of providing different learning paths in closed-lab courseware
The capability of a CBI program to provide different learning paths depends on its continual assessment of learners' performance. A question immediately follows: 'Then how does a lab-based courseware adapt to different learners if it does not assess learners' performance at all?' Control of a learning sequence in a lab-based software is determined by the lab manual.
not by the software. Since the software does not control the learning sequence, it is unreasonable to expect it to provide different learning paths. It is impractical, if not totally impossible, to prepare different sets of procedures in the lab manual for different students either. This dilemma has led us to regard the issue of individual differences as irrelevant to the design of lab-based software for the time being; however, we do believe that any teaching method, including lab-based teaching, can be effective only if it provides some solution to the fundamental problem that students do not all learn alike. In other words, lab-based software has to somehow find a way in dealing with this subtle question. We have not heard of any discussions about this issue so far, but it is indeed worthy of further investigation.

4 Guidelines for developing lab-based software

The issues that we have discussed so far provided the basic groundwork for understanding lab-based software. In this section we propose some practical guidelines for developing good lab-based software.

4.1 Selection of suitable topics

In Section 2.1 it has been mentioned briefly that closed labs should be used mainly to support lectures on those topics that are found most difficult to understand by the students. The difficult topics usually involve concepts that are abstract and/or complicated. It is the job of a lab-based software to help students organize and visualize their imagination about a certain concept. For example, the majority of students who experimented with our SimRECUR and SimLIST software agreed that without aids like these they could hardly visualize program dynamics of recursion and the linked lists. On the contrary, designing lab-based courseware for topics that do not involve difficult concepts will not only waste lab time, the courseware itself is destined to fail. We consider one of our courseware, NUMBER (Figure 2) to be a less successful lab courseware because of this reason. Such topics as conversions between different number systems and the arithmetic operation of binary numbers are tedious at best, they are not really difficult to understand by average students. It turned out that the NUMBER software may be more suitable for open-lab exercises.

4.2 Extraction of information

Designers of lab-based courseware have to be especially careful in identifying the essential part of the content knowledge in a topic to be covered by the courseware. To put it in another way, the content covered should be both sufficient and necessary. Avoid putting more content knowledge than is necessary in a lab-based courseware. Failing to present the necessary information in the courseware is an even greater mistake. For example, in designing a courseware for teaching looping constructs of a programming language, the designer should be able to realize that the observation of changes in loop control condition(s) during iterations of the loop body and at what point the condition is tested in each iteration is crucial to students' understanding. Simply laying out some sample programs containing loops together with the output it produces will not result in a successful courseware. In order to extract the essential knowledge from a topic, it is necessary for the courseware designer to conduct interviews with experienced teachers in order to know what learning difficulties students typically have about a certain topic. Giving tests to a selected group of students and analyzing the test results will help too.

Several high-school computer teachers who participated in our research projects pointed out they have encountered difficulties drawing students' attention to a particular portion of the application software which they were teaching. Some students just seemed to enjoy more exploring freely than being guided step by step by the instructor. They usually would not be stopped unless they got stuck somewhere. Provision of only the necessary information on the computer screen will not only avoid distracting the curious learners but simplify the software development process.

4.3 User interface design

The basic requirement for a good user interface is user-friendliness. For computer software, the term usually means that information presented on the computer screen look pleasant and that the objects to be manipulated by learners, including on-screen menus, the icons, buttons, dialog-boxes, etc., are easy to use. Provision of on-line help is another feature that contributes to ease of use. For instructional software, an additional criterion of good user interface design is how the material is presented to enhance learning effects. The guiding principles provided by CBI textbooks for user interface design apply equally well to lab-based software. However, a number of special considerations should be given to the interface design of lab-based software.

1. It is often the case that several lab sessions will be scheduled for a course in a semester. The set of courseware used in a series should be as consistent as possible in their user interfaces. Unnecessary variations in interface design of individual software system will only slow down the learning process or even confuse the students.

2. Consistency in designing the computer screens means 'implementing same thing in the same way.' The set of courseware used for a single course may not be designed by the same designer. It is desirable that the courseware designers agree upon certain conventions for laying out screen objects, especially the objects that are to be manipulated by learners to perform lab activities. For example, the designers of two software systems for teaching sorting and searching algorithms respectively may agree on how computer screens will be divided into certain number of sections to accommodate necessary items, and that each should provide the same four control functions for students to trace an algorithm, namely run, step, halt, and back up. Furthermore, the four functions are to appear as individual push-on buttons instead of items of a pull-down menu. The designers may even agree on the shape, the color, the position of each button.

3. User interface of lab-based courseware means more than layout of material and control functions on the computer screen. The lab manuals and the worksheets also contain information for the learners and should be considered as parts of the user interface too.

4. Consistency in designing lab manuals and worksheets centers on its necessary content and its format. It is important that the lab-based courseware designers establish a set of formatting instructions for manuals and worksheets. For example, the formatting instructions for a lab manual may specify that each manual must include the following major sections in the prescribed order: the name of the laboratory, the objective, hardware and software system requirements, the pre-lab tutorial, the experiment procedure, and the post-lab discussion. The instructions may also give guidance on the detailed format for
each section as well as the principles to be observed in designing the more critical sections, such as the experiment procedure and the list of questions for post-lab discussion.

4.4 Lab manuals and the worksheets
The software, the lab manual, and the worksheet of a lab-based courseware have great mutual dependency on each other, and should, therefore, be designed and developed in parallel. According to our experience, a lab-based courseware designer has often been more interested in designing and coding the software than preparing lab manuals and the worksheets. Unless reminded repeatedly, they tended to work on the software all the way till it is completed, and only then did they start thinking about the lab manual and the worksheets. This often resulted in the need to modify the software in order to accommodate certain meaningful lab activities in the manual. The designer should always consider the software, the lab manual, and the worksheets as equally important, and should avoid treating the manuals and the worksheets as dirty work.

5 Concluding remarks and directions for future research
In this paper we discussed the characteristics exhibited by courseware designed for CS closed laboratories. We then looked into the instructional factors relevant to the design of lab-based courseware. We finally proposed some guidelines with regard to the selection of suitable topics, presentation of information, and the design of user interface for lab-based courseware. The points raised in this paper may seem immature and may very well be debatable. However, what we hope is that more researchers will be prompted to share their experiences and findings about the development processes of their lab-based courseware. We agree with McAlpin's comments in [6] that 'poor dissemination of information about methods and software has resulted in considerable duplication of effort.' Such a situation can be improved in a number of ways. In the technical respect, we need to establish one or more models for designing lab-based courseware. These models should not only be deeply rooted in formal instructional methodologies and learning theories, but provide practical procedures and guidelines for developing effective lab-based courseware. It is our wish that we could finally propose such a model in the future.

We should also address other non-technical issues concerning the development of lab-based courseware. Currently the lab-based courseware products are used mostly in the institutions of origin. It is often difficult for other researchers to access such courseware across institutional boundaries. It would be most desirable if some organization, such as ACM SIGCSE or SIGCUE, could undertake the task of installing a laboratory courseware library of existing lab-based courseware and provide the courseware free or for a nominal charge to the researchers in the field. The establishment of such a library will be a first step towards forming a computer science laboratory infrastructure advocated by Birch [2], and it will give further impetus to the incorporation of closed laboratories in computer science education. Before the ideal of such a library becomes realizable, an exhaustive survey of existing lab-based courseware periodically by senior researchers in the field will have great contributions to this research field too.

References

Acknowledgment
This research has been funded by the National Science Council of Taiwan, the Republic of China, under the grant numbers NSC 84-2511-S-003-024, NSC 84-2511-S-003-078, and NSC 85-2511-S-003-058.