COMPUTER-ASSISTED INSTRUCTION and INTELLIGENT TUTORING SYSTEMS

Shared Goals and Complementary Approaches

Edited by
JILL H. LARKIN
RUTH W. CHABAY

LEA

G434
C738

9461189

COMPUTER-ASSISTED INSTRUCTION and INTELLIGENT TUTORING SYSTEMS:

Shared Goals and Complementary Approaches

E9461189

Edited by
JILL H. LARKIN
RUTH W. CHABAY
Carnegie Mellon University

[illegible]图书馆藏书

LEA LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1992 Hillsdale, New Jersey Hove and London

Carol Scheftic

Editor for Content and Style
during the crucial early period of the book's development

Copyright © 1992 by Lawrence Erlbaum Associates, Inc.
All rights reserved. No part of this book may be reproduced in any form, by photostat, microform, retrieval system, or any other means, without the prior written permission of the publisher.

Lawrence Erlbaum Associates, Inc., Publishers
365 Broadway
Hillsdale, New Jersey 07462

Library of Congress Cataloging-in-Publication Data

Computer-assisted instruction and intelligent tutoring systems : shared goals and complementary approaches / edited by J.H. Larkin, R.W. Chabay.
p. cm.
Includes bibliographical references and index.
ISBN 0–8058–0232–0.—ISBN 0–8058–0233–9 (pbk.)
1. Computer-assisted instruction. 2. Intelligent tutoring systems. I. Larkin, J. H. (Jill H.) II. Chabay, R. W. (Ruth W.)
LB1028.5.C5288 1991
371.3'34—dc20 91–15081
CIP

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

COMPUTER-ASSISTED INSTRUCTION and INTELLIGENT TUTORING SYSTEMS:

Shared Goals and Complementary Approaches

TECHNOLOGY IN EDUCATION SERIES

Edited by
Raymond S. Nickerson

Nickerson/Zodhiates • Technology in Education: Looking Toward 2020

Larkin/Chabay • Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches

Bruce/Rubin • Electronic Quills: A Situated Evaluation of Using Computers for Writing in Classrooms

此为试读，需要完整PDF请访问：www.ertongbook.com

Contents

Introduction

Jill H. Larkin and Ruth W. Chabay, Editors
Center for Design of Educational Computing, Carnegie Mellon University

THE PURPOSE OF THIS BOOK

Two groups of individuals share a vision that computers can provide excellent instruction for large numbers of students. The first of these groups we call developers of computer-assisted instruction (CAI). This group consists predominantly of experienced teachers and educational researchers, with strong backgrounds in the subject matter of their programs. The second group we call developers of intelligent tutoring systems (ITS). This group consists predominantly of researchers in cognitive psychology and computer science who develop principles of learning and apply them in instructional programs.

Unfortunately, these groups have had few vehicles for sharing ideas or programs. Different backgrounds and settings have meant reading different journals and attending different conferences. The purpose of this book is to foster a mutual understanding of shared issues and complementary approaches so as to further powerful educational applications of computing.

The following pages first summarize the complementary, but distinct, approaches of CAI and ITS, and then discuss shared issues toward which these complementary approaches are directed.

COMPLEMENTARY APPROACHES

Central Aims

The aim of CAI programs is to address existing needs of particular groups of students. The CAI developer wants to produce the program that has the best chance of teaching effectively and applies to this end all available experience and expertise. CAI programs are specific and hand-crafted for the domain, topic, and students addressed. With these specifically tailored programs, CAI pushes the frontier of the best that can be done with current technology and imaginative techniques.

In contrast, the aim of ITS developers is to implement in programs a set of instructional principles sufficiently general to provide effective instruction for a variety of teaching tasks. ITS programs are strongly rooted in research on the psychology of learning. With large programs that provide instruction for many tasks (e.g., a significant part of a course), ITS pushes the frontier of knowledge about what general instructional techniques work and why.

Instructional Models

CAI programs do not follow a single theoretical model of instruction. In many CAI programs, the instruction emulates (in a form appropriate to the computer medium) interactions that might occur between a student and an excellent teacher. Other programs attempt to create an engaging, motivating environment that encourages purposeful exploration in a domain. A rich diversity of environments and problems is often a goal in CAI, and a suite of programs developed for a single course may vary significantly in goals, tasks, and style.

CAI programs reflect their developers' experienced beliefs about good teaching and good design for the computer medium. Some of the most interesting programs derive from developers' intuitions about activities well-suited to the medium, rather than from traditional instruction in the domain. Learning goals can be implicit in CAI, and activities may or may not be explicitly related to the tasks the student will be expected to perform after instruction.

ITS programs, in contrast, contain an explicit computer-implemented model of instruction. It is this model, and not the hand-crafted code of the developer, that determines how the program responds to the student. This model of instruction consists of two parts: (a) a *performance model* capable of performing the tasks the student is learning to perform, and (b) a *teaching model* that compares the student's actions with those of the performance model, and determines what (if any) reactions the ITS will provide to the student.

The performance model usually consists of a large set of fine-grained inference rules (although other techniques may be used in conjunction with these rules). Each rule consists of actions together with conditions under which these actions should occur. It is important that the rules be small in scope, because then they can be combined flexibly, in different ways, to do different activities. When an ITS is running, these rules are not invoked in any fixed order. Instead, at each point in time, the program searches for the rule most likely to contribute useful information to the current situation. This rule-based program architecture has had repeated success in building computer programs that apply large amounts of knowledge flexibly to a variety of tasks.

The teaching model in an ITS consists of engaging the learner in a reasoning task, and comparing the student's actions with the set of actions that can be generated by the performance model. This comparison lets the ITS identify when the student does something either wrong (according to domain knowledge) or useless in pursuit of the current task. The teaching model then determines what (if any) advice is to be offered. Furthermore, the tasks given to the student are designed so that (a) all inference rules in the performance model are required to solve the full set of tasks, and (b) each inference rule is applied in several contexts. Thus, the student has multiple opportunities to practice and acquire the knowledge represented by each rule.

In summary, CAI programs reflect experience in teaching in a particular domain, and consist of varied activities designed to help students increase domain knowledge and apply that knowledge in different contexts. ITS programs have an explicit model of the knowledge required for a domain of tasks, and the activities they include provide systematic practice in using all relevant knowledge.

Program Structure

CAI programs are interaction centered, reflecting the CAI model of instruction in which the computer is an interactive medium for instruction characteristic of excellent teachers. CAI programs are therefore built to provide a specific kind of student interaction with the computer screen. These programs start with a vision of this interaction, and then programming is done to make that vision real. CAI developers see the computer screen as a programmable interactive communications medium.

Single CAI programs are usually relatively brief (although some can be used in many ways) and deal with only a few aspects of a domain (e.g., comparing the size of fractions). A set of related programs, carefully coordinated by their developer(s), can provide instruction throughout an entire course.

In contrast, *ITS programs are knowledge centered*, reflecting the explicit model of performance knowledge on which they are based. Interaction with a student consists of the model sending a message to the interface which in turn uses its own knowledge of graphics and layout to convey the information to the student, and to convey student input back to the central model.

ITS programs are typically comprehensive, with one program, in a single format, providing most or all of the instruction for a large fraction of the material in a course. The knowledge encoded for early lessons is re-used when appropriate for later lessons, much as the student is intended to use knowledge acquired earlier, along with new knowledge to address more complex tasks. An

此为试读，需要完整PDF请访问：www.ertongbook.com

ITS systematically teaches a body of knowledge by providing multiple and varying opportunities for the student to learn each inference rule in the performance model.

Necessary Experience

Because CAI is interaction centered and because it takes time to learn the sensitive use of a new and difficult medium (e.g., film and television as well as the computer), many of the best CAI programs are written by people with years of experience. The authors in this book, for example, have up to 20 years experience in the computer medium.

In particular, three of the CAI chapters reflect work which began in the PLATO[1] instructional computing environment originating at the University of Illinois. This environment provided a graphics screen with nearly twice the number of pixels as the standard Macintosh, a touch interface, and the ability to back-project microfiche—all available in the early 1970's! The size of the PLATO system allowed very large courses to require several hours of computer work each week. Thus new programs could accrue hundreds or thousands of hours of student use each semester.

In contrast, ITS is based on efforts to represent explicitly the knowledge humans use in performing a set of tasks. Building such programs is an extremely difficulty task, and stresses existing principles and techniques in both psychology and computer science. Therefore, developers of ITS are usually researchers in psychology or computer science, and have extensive experience in both fields. In particular, they draw on about 20 years of research on detailed processes of the human mind, and ways of characterizing these processes through computer-implemented models.

Computer Implementation

The aim of CAI is to provide practical instruction, consisting of interactive programs that teach effectively. Therefore CAI programs are developed under heavy computational and other constraints (eloquently described in the chapter by Brackett). They must respond rapidly (to support interaction and varied graphic interactions). Without graphics and fast interaction, no CAI program described in this book could exist in anything like its current form. Yet these programs must run on affordable and widely available machines (low end Macintoshes and PCs, and the educationally ubiquitous Apple II), machines with severe memory and speed limitations.

[1]PLATO® is a registered trademark of The Roach Organization, Inc. The PLATO system is a development of the University of Illinois.

CAI programs are generally algorithmic in structure (i.e., describable by a flow-chart) with user input often determining branching. Programs are organized around the screen display, and interactions with students center on the display, which changes in response to various inputs from the student. Algorithmic programs can be faster and simpler than those with more complex architectures. In CAI, the most-used languages are algorithmic languages, e.g., versions of BASIC, PASCAL, and assembly language, as well as descendents of TUTOR (the PLATO programming language).

In contrast, ITS programs exploit models of knowledge and teaching. Because the human mind is complex, computational models of its functioning are complex. Implementing them requires powerful machines and the most sophisticated techniques of computer science research.

ITS programs can not be algorithmic because humans have abilities far more varied than those of any algorithmic program. Instead ITS programs have a logic based on repeatedly applying knowledge (encoded as rules) to react to a current situation. Furthermore, content of this knowledge consists not of numbers, but of symbols (words, phrases, rules). ITS systems are therefore almost always written in languages that support processing of symbols and lists (e.g., LISP, Prolog) and their rule-based extensions (e.g., GRAPES, OPS5). ITS systems use extensively techniques of "artificial intelligence," that is techniques for building large programs that can incorporate large amounts of knowledge, and use it flexibly. For efficiency, ITS programs are sometimes written (or rewritten) in C, one of the most powerful modern algorithmic languages.

Implications

The differing aims, constraints, and instructional models imply the following substantial differences between CAI and ITS programs.

CAI programs are hand-crafted using deep knowledge of the domain, the students, and the computer medium. In CAI, the interface is intuitive, and using it is part of the instructional process. Good CAI has clarity and charm. Despite their algorithmic architecture, the clever hand-crafting of interactions gives these programs the feeling of freedom of interaction. Good CAI typically has been used for many thousands of hours by varying students, although quantitative studies of its effectiveness are rare.

In contrast ITS programs are the product of an applied science. They are based on, and explicitly include, psychological principles of learning. The instruction aims to provide systematically opportunities for learning all knowledge in a carefully defined domain. These programs are often meticulously tested with student groups, and their effectiveness can be dramatic (although there are few demonstrations of continued utility in widespread use).

SHARED PRINCIPLES

The chapters in this book, each describing the development and nature of one or more instructional programs, make concrete the contrasts discussed above. However, they also illustrate a rich set of largely shared design principles for good educational computing. The following paragraphs summarize these shared principles, which appear as continuous themes throughout the chapters in this book.

Engage the student in active work on central tasks

For example, Corbett & Anderson's principles of intelligent tutoring include providing instruction in the context of active problem solving. Chabay and Sherwood, summarizing their experience-based guidelines for CAI, give the mandate "Ask, don't tell," and go on to discuss techniques for designing programs that consistently engage the student in active work. This principle is reflected by every instructional program described in this book. Students do not passively read screens of text, or simply watch graphic simulations.

Use the interface to suggest appropriate reasoning

Displays and interfaces have long been a primary focus of CAI. In particular, Dugdale provides an excellent description of the intricate interaction between the domain being taught and the intricate displays that teach it. Only recently have ITS developers given significant attention to the difficult task of designing principled and effective graphic interactions. The chapters by Reiser et. al, and by Lesgold et. al., discuss this work.

Center work on appropriate reasoning

Most of the time the student should be working smoothly and successfully on important tasks. In Dugdale's *Green Globs*, for example, students are free to write whatever equations they like, with the aim of producing a graph that will pass through "globs" placed on the screen. Any engagement in the program means thinking about the relation between equations and their graphs. The actual graph that does appear is itself immediate feedback on whether the student's thinking about this relation was accurate or not. In the ITS world, relevant, successful, active thinking is achieved by tracing inference rules used by the student, and by intervening with a correction or hint if the student moves more than a step or two away from an appropriate reasoning sequence. This does not mean the student can solve a problem in just one "correct" way—the program's inference rules always attempt to support any correct solution, and often to provide strategic advice on incorrect reasoning.

Provide prompt feedback focusing attention on erroneous thinking

For example, Dugdale's elegant CAI programs are designed so that the interface gently prohibits many irrelevant actions, and the results of relevant actions are immediately apparent. All of the ITS programs are designed to track the student's reasoning and to give increasingly detailed feedback when the student does something wrong or irrelevant. In particular, the chapter by Reiser, et., al., discusses the use of knowledge in the ITS to formulate this feedback.

Adapt instruction to individual student knowledge

Corbett & Anderson describe how ITS programs can adapt through selecting the inference rules required by the task—a small set at first, more later on. In contrast the ITS *SHERLOCK* (Lesgold, et. al.) provides a single set of problems, but uses hints to suggest inferences the student finds difficult. Dugdale's CAI adapts to students' knowledge by letting students set their own tasks, within a cleverly designed environment where involvement in any task is likely to increase skill and ability to design and solve more complex problems. Many CAI programs (e.g., Culley's earlier grammar programs), provide the student with unlimited practice, together with information on success rates. Thus students with varying initial capabilities can readily use the program as a tool to build skill.

VALUE OF INTERACTION

Although the shared instructional principles discussed above are, of course, implemented quite differently in ITS and CAI programs, there are also the beginnings of joint implementations. For example, the chapter by Culley describes how a CAI developer began explicitly to need the techniques used in ITSs. Sack and Soloway, ITS developers, discuss both their needs for the evaluation techniques of educational research, and for a program structure that provides greater student interaction.

The purpose of this book is to acquaint both ITS and CAI developers with a broad range of approaches to their common vision of broadly effective educational computing.

ACKNOWLEDGEMENTS

We wish to acknowledge, with great appreciation, the central contributions of the following individuals and organizations:

The James S. McDonnell Foundation which provided funding for the conference, held in March, 1988, on which this book is based, and for preparation of the manuscript. John Bruer, President of the Foundation, provided

intellectual and moral support to complement the financial support of his organization. During initial planning for the book and conference, Jill Larkin was supported by a fellowship from the Guggenheim Foundation and by a National Science Foundation Visiting Professorship for Women.

Neither the conference nor the book would have been possible without the resources and working environment provided by Carnegie Mellon University and its Center for Design of Educational Computing. Additionally the Software Engineering Institute provided excellent facilities for a technologically complex conference.

Stanley Smith of the University of Illinois challenged and enlivened the workshop with his demonstration of innovative chemistry programs, which allow students to see and analyze real chemical phenomena using interactive videodisc technology.

Carol Scheftic served ably as the central contact with authors and as editor during the first difficult phases of manuscript preparation. Carol Scheftic and Ruth Chabay also took full responsibility for the conference leading to this book, while Jill Larkin managed to attend the sessions despite pneumonia.

Alice Huston patiently and meticulously read all of the ITS chapters, giving invaluable suggestions for making these chapters readable to an audience outside of the computer science community.

Finally, Mary McCallum provided patient expertise in proofreading, correcting endless versions of manuscripts, tracking down detailed information, and transferring materials (physically and electronically). She cheerfully took on endless extra tasks both contributing directly to this book and giving time for the editors to edit — a task more difficult than either of us had imagined.

Jill H. Larkin and Ruth W. Chabay
Pittsburgh, October, 1991

SUMMARY OF CHAPTERS AND PROGRAMS

Ch. No.	*Author*	*Program Type*	*Age Level*	*Subject matter*
1.	Dugdale	CAI	Pre College	Mathematics
2.	Culley	CAI /ITS[2]	University	Latin
3.	Corbett & Anderson	ITS	University	Programming in LISP
4.	Reiser et al.	ITS	University	Programming in LISP
5.	Chabay & Sherwood	CAI	All Levels	Design issues in CAI
6.	Brackett	CAI	Pre College	Slide presenter & scientific reasoning
7.	Lesgold et al.	ITS	Military	Electronic device troubleshooting
8.	Sack & Soloway	ITS[2]	University	Programming in PASCAL

[2]These two chapters describe programs which are not "tutors", but which use techniques of artificial intelligence.

此为试读，需要完整PDF请访问：www.ertongbook.com