UNIVERSITY OF JOENSUU
COMPUTER SCIENCE
DISSERTATIONS 1

MARJA KOPPONEN

CAI IN CS

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Science of the University of Joensuu, for public criticism in Auditorium M4 of the University, Yliopistokatu 7, Joensuu, on April 11, 1997, at 12 noon.

UNIVERSITY OF JOENSUU
1997

Julkaisija Joensuun yliopisto
Publisher University of Joensuu

Toimittaja MarttiPenttonen
Editor

Vaihto Joensuun yliopiston kirjasto, vaihdot
PL 107, 80101 JOENSUU
Puh. 013-251 2672, telefax 013-251 2691

Exchange Joensuu University Library, exchange
P.O.Box 107, FIN-80101 JOENSUU
Telefax +358 13 251 2691

Myynti Joensuun yliopiston kirjasto, julkaisujen myynti
PL 107, 80101 JOENSUU
Puh. 013-251 2652, 251 2662, telefax 013-251 2691
Email:lavikainen@joyl.joensuu.fi

Sale Joensuu University Library, sale of publications
P.O.Box 107, FIN-80101 JOENSUU
Telefax +358 13 251 2691
Email:lavikainen@joyl.joensuu.fi

ISSN 1238-6944
ISBN 951-708-506-0
UDK 681.3
Computing Reviews (1991) Classification: K.3.1, K.3.2
Yliopistopaino
Joensuu 1997

CAI in CS

Marja Kopponen

Department of Computer Science
University of Joensuu
P.O.Box 111, FIN-80101 Joensuu, Finland
Marja.Kopponen@joensuu.fi

University of Joensuu. Computer Science, Dissertations 1
Joensuu, April 1997, 99 pages
ISSN 1238-6944, ISBN 951-708-506-0

Keywords: computer-aided instruction (CAI), use of CAI, evaluation of CAI, COSTOC

By computer-aided instruction (CAI) we mean the use of a computer application for instructing a specific subject of a domain. The essential question is what CAI applications are good and why. In order to be able to evaluate a CAI application it is necessary to fix the domain. The domain of this work is computer science at the university level.

Evaluation of CAI applications is a complex process in which several perspectives have to be considered. We developed evaluation criteria that consist of four parts, namely, domain-based criteria, instructional criteria, user interface criteria, and pragmatic criteria. Our domain-based criteria focus on evaluating the course contents and their relevancy to the instructional aims of the CAI course. We examined several human learning theories in order to find the most appropriate one to be the basis of our instructional criteria, which concentrate on evaluating the educational support. User interface criteria and pragmatic criteria focus on evaluating the implementation of the user interface and the practical matters, such as hardware, software and human resources, respectively.

We designed an analysis method based on our criteria for testing the evaluation criteria in practice. The analysis method was applied to a collection of CAI courses on computer science, called COSTOC (COmputer Supported Teaching Of Computer science).

The analysis indicated that our criteria worked properly. The main results were that the contents of a CAI course have to be designed by an expert of the domain and that the instructional support should be based on human learning. Further, authoring tools should support CAI authors by offering instructional advice. In addition, authoring CAI courses on computer science requires some special properties of the authoring tool, such as possibilities to speak mathematical language, to animate abstract structures, and to write and execute pseudo or programming code.

Preface

This thesis is the result of research carried out in the Department of Computer Science at the University of Joensuu during the years 1994-1996, and in the Laboratory for Computer Aided Learning, Swiss Federal Institute of Technology during the academic year 1994-1995.

I am deeply grateful to my supervisor, Professor Martti Penttonen, for his invaluable advice and comments. His endless encouragement gave me strength to pass the difficult moments and his delightful linguistic jokes cheered me up.

To my other supervisor, Associate Professor Jorma Enkenberg, I express my sincere thanks for tirelessly guiding me in the world of Educational Sciences.

I wish to thank Associate Professor Jorma Sajaniemi and Assistant Markku Tukiainen for providing their valuable help and knowledge concerning user interfaces. I especially want to thank Markku for all those discussions we have had during these years. I also want to thank the whole staff of the Department of Computer Science at the University of Joensuu for support and empathy during this work.

I earlier worked and studied in the Department of Computer Science and Applied Mathematics at the University of Kuopio. I express my special thanks to the staff of that department, especially Acting Associate Professor Seppo Lammi. To my dear friend, Ph.Lic. Virpi Kasurinen, I express my warmest thanks for the moments we have shared during our friendship.

I want to express my warmest thanks to Professor Frédéric de Coulon for the opportunity to work in his laboratory during my stay in Switzerland. I also want to thank Dr. Eddy Forte for his valuable advice and B.Sc. Guy Delafontaine for all the care during my visit.

To my official referees, Professor Erno Lehtinen and Docent Matti Linna, I am greatly indebted for their careful review of the manuscript and valuable comments on it.

Special thanks are due to my colleague M.Sc. Stephen Eriksson-Bique for revising the language of this manuscript.

The financial support of the Centre for International Mobility, the Department of Computer Science at the University of Joensuu, the East Finland Graduate School in Computer Science and Engineering, the Emil Aaltonen Foundation, the Finnish Cultural Foundation, the Finnish Konkordia Foundation, and the Swiss Confederation is gratefully acknowledged.

Finally, I owe my dearest thanks with a hug to my sweetheart Markku for his love and support. Without him my thesis might never have become completed.

Abbreviations

Abbreviation	Description	Chapter or section where defined or first time referred to
ACM	Association for Computing Machinery	2.3
ACT*	Adaptive Control of Thought*, * belongs to the name	3.1.2
AIDA	Advanced Instructional Design Associate	3.2.1
CAD	Computer Aided Design	4.2
CAI	Computer-Aided/Assisted Instruction	1
CAL	Computer-Aided/Assisted Learning	1
CBE	Computer-Based Education	1
CBI	Computer-Based Instruction	1
CBT	Computer-Based Training	1
CDT	Component Design/Display Theory	3.1.4
CMI	Computer-Managed Instruction	1
COSTOC	COmputer Supported Teaching Of Computer science	1, 5.2
CRI	Criterion Referenced Instruction	3.1.3
CTGV	Cognition & Technology Group at Vanderbilt	3.1.3
DDL	Data Description Language	5.7.3
DML	Data Manipulation Language	5.7.3
GAIDA	Guided Approach to Instructional Design Advising	3.2, 3.2.1
GOMS	Goals, Operators, Methods and Selection rules	3.1.2
GPS	General Problem Solver	3.1.2
PLATO	Programmed Logic for Automatic Teaching Operations	1
TICCIT	Time shared Interactive Computer-Controlled Information Television	1

1 Introduction

Introduction

Since 1920's researchers have been looking for different ways to utilise teaching machines, auto-instructors and, later on, computers in instruction. Despite that, computers are only seldom used systematically in teaching. There exist many reasons for that, such as unawareness of human learning, the complexity of the design process of computer-aided instruction (CAI) applications, and the fast development of computer technology, which has caused problems since CAI materials designed for certain computer configurations become obsolete in a few years. In the early days of computer-aided instruction there was a dream that human teachers will someday be replaced by computers. This dream is nowadays rejected because it has become clear that computers are not capable to take care of the role of a human teacher since, firstly, human contact is very important and, secondly, human-like features, such as heuristical problem solving and concluding, should progress extremely within computers, which hardly happens in near future. Therefore, human teachers will keep on teaching, maybe not the way they have always taught but using computers as tools.

Computer-aided instruction can be defined concisely as a teaching method in which a student interacts with a specialised computer application designed to instruct certain skills and knowledge. A more extensive definition includes the use of computers in all study tasks, such as students using computer applications as tools (e.g., text processors, spreadsheet calculators), teachers preparing lesson materials (e.g., lecture notes, presentation materials, software demonstrations, CAI applications), or school administration personnel organising educational administration on computers (e.g., curriculum design, student files). Depending on the situation, several abbreviations can be used, such as CAI (computer-aided/assisted instruction), CAL (computer-aided/assisted learning), CBE (computer-based education), CBI (computer-based instruction), CBT (computer-based training), and CMI (computer-managed instruction). These abbreviations have different shades of meaning, though they all refer to

use of computers in education. This study concentrates on CAI in its concise meaning, that is, the use of computer applications in instruction.

CAI is considered to be based on Pressey's teaching machines dating back to the 1920's. According to Sloffer (1996), these teaching machines presented questions or problems to students who had to respond by writing an answer or pressing a certain button. Students were informed about the correctness of the answer and occasionally given the explanation why it was right or wrong. An account of responses was often kept. In the 1950's Skinner introduced his version of a teaching machine and programmed instruction which were based on his theory of operant conditioning. According to Li (1996), programmed instruction had important long-term effects on the evolution of educational technology; namely, it had a strong influence on the development of systems approach, and it introduced the early ideas of individualisation of instruction. A decade later teaching machines were considered to be obsolete since reseachers perceived that as a teaching tool the machines are not as important as the programs. The concept of CAI was introduced in the 1960's.

Two important projects in the history of CAI are PLATO (Programmed Logic for Automatic Teaching Operations) and TICCIT (Timeshared Interactive Computer-Controlled Information Television). These two projects were competing with each other to become the national automated instruction system in United States of America. Even though both of them were based on mainframe computers, the rest of the technical solutions differed a lot from each other. Neither of them became predominant since the expected fall in installation and utilisation costs did not happen. Nevertheless, both of them served as a basis for other instructional systems.

The built-in educational properties are the essence of CAI applications. These properties include clearly stated learning aims, methods to reach these aims, and testing whether these aims are achieved. In addition, CAI applications support and guide students' proceeding in a CAI application. These properties separate CAI applications from other computer applications, such as multi- and hypermedia applications and expert systems, which all have been used for instructional purposes. Since the educational properties are usually missing from these other applications, the instructional use of them differs noticeable from the use of CAI applications. Thus, CAI applications can be used as tutors, while the other computer applications can be considered as tools.

Traditionally teaching has been a situation in which a teacher talks in front of students. Teaching this way is strongly teacher-centered but many times it has been the only possibility. Reasons for that could have been the lack of teaching materials, teachers, classrooms, etc. Also the earlier learning theories assumed that students are like empty baskets and teacher's task is to fill them with information. These kind of theories are not valid anymore because research has revealed that human learning is more complicated. This observation has affected the teacher's role in instruction.

Researchers of human learning have presented lists of educational or instructional tasks, which should provide the necessary conditions for learning. These tasks are good to remember while designing instruction and selecting appropriate media. As an example, Gagné and Briggs (1974) have presented the following nine instructional events:

1. *Gaining attention* means that students' interests in teaching subject are awakened. This can be done, for example, with questions or demonstrations.
2. *Informing the learner of the objective* gives students the learning aim. This phase is not necessary if the aim is obvious.
3. *Stimulating recall of prerequisite learned capabilities* helps students to combine their prior knowledge to new information. Stimulation can be done asking recall questions.
4. *Presenting the stimulus material* is closely connected to the learning aim. It is necessary that stimuli must support students to reach the aim.
5. *Providing learning guidance* is an event in which students are guided to the direction of the correct answer or performance. The form of learning guidance can be, for example, questions or hints proposed during instruction.
6. *Eliciting the performance* means that students show what they know or are able to demonstrate their skills.
7. *Providing feedback* is telling students the correctness of their performance. In some cases this is not necessary if there exists 'automatic' feedback, such as in throwing darts the result is immediately clear.
8. *Assessing performance* focuses on the reliability and validity of students' performance. A couple of appropriate questions that force students to apply their new knowledge should convince the teacher that students' performance is valid.
9. *Enhancing retention and transfer* means systematic reviews and practicing of skills.

A teacher does not have to strictly follow these steps described above. A better result is often achieved when these steps have been used to form a fluently proceeding, well-adjusted unity in which these demands have been taken into account.

Within instruction, teachers are responsible of the totality. They decide what to teach and how to teach, and they take care of the needed material. It is assumed that teachers apply some learning theory while designing the lessons. This is something that computers cannot do by themselves. Furthermore, teachers are human beings with their gestures, expressions and intelligence. They are able to think, to make heuristic conclusions and use body language. For example, students easily see if teachers are motivated and interested in topic they are teaching. That is likely to increase students' motivation and improve learning results. Likewise, teachers are able to react to students' behavior, for example, when they seem to be confused or have problems.

Computers have no human properties which is the most important reason why they cannot replace human teachers. Instead, computers should be used as tools in instruction because they have many valuable properties, such as availability in a sense that they are cheap, relatively high speed, tirelessness, large memory capacity, and ability to animate graphics. It would be wise to use these properties whenever it is appropriate.

One of the strongest advantages of computers is their ability to present animated graphics. According to Kiser (1987), this unique feature permits designing of computer lessons that instruct in a way that static formats cannot. Richards and Fukuzawa (1989) have pointed out that seeing shapes, forms, and symbols manipulated in a concrete way is more effective than trying to visualise an abstract model in one's mind. Baek and Layne (1988) have reported that well-designed computer lessons with animation have improved student scores more than presentations with graphics and text or text alone. Since animation is a strong effect it must be used wisely. It overrules the main subject if it is used too much or in a wrong place.

Computers have a great memory capacity which can be used for storing large amounts of information. They are also fast in certain tasks. Rapidity is useful, for example, when searching information from diverse sources for writing an essay, or solving mathematical problems containing time-consuming routines.

Tirelessness is valuable, for example, when practicing vocabulary or similar tasks. Students can repeat difficult things as many times as needed. They are also able to study as long as they like and continue at their own pace. In classroom usage, proceeding with individual speed is valuable because students will not become frustrated studying at an appropriate speed for them.

Computers treat students equally ignoring their sex, color and social class. There are no bad days for computers, that is, instruction is not depending on feelings or other human features. Furthermore, computers can be equipped with unusual peripherals, such as touch screens or special keyboards, to enable the usage for disabled students. For many seriously disabled persons a computer offers a possibility to communicate with other people and to study.

1.1 The goals of this work

This work focuses on designing criteria for evaluating CAI applications. The study is based on human learning theories and on analysis of some CAI applications on computer science; consequently, the relations between learning theories and CAI applications are examined.

Many studies concerning the usage of CAI in teaching computer science have been conducted. However, little attention has focused on how computer science should be instructed. Therefore, the goals of this research are to find out what kind of CAI applications are designed and should be designed for teaching computer science, how human learning theories are applied and should be applied to these applications, and does there exist some special properties that are required for designing CAI applications on computer science.

CAI authoring tools have a strong influence on CAI applications since their properties set the limits on CAI application design process. Therefore, it is important to know what kind of tools have been used for authoring CAI applications on computer science and how the resulting applications look like. A short review on CAI authoring tools and their applications

on computer science is presented in the second chapter. A more detailed examination of some CAI applications on computer science is presented in the fifth chapter.

The design of CAI applications should be derived from students' needs, rather than technical aspects. Therefore, human learning theories should be the basis for the design process. Learning theories related to CAI are shortly introduced in the third chapter. In addition, there are presented descriptions of four human learning theories of current interest in a form of a case study.

Based on the experiences from CAI authoring tools, existing CAI applications on computer science and human learning theories novel evaluation criteria for CAI applications will be developed. The criteria will include both educational requirements and requirements of the subject domain and user interface design, as well as the pragmatic demands. The fourth chapter presents these criteria for evaluation of CAI applications on computer science.

The description of the analysis method based on the criteria is presented in the fifth chapter. CAI applications called COSTOC were chosen to a detailed analysis. These COSTOC courses are introduced in the fifth chapter, as well as the results of their detailed analysis.

2 CAI authoring tools and their applications in computer science

CAI authoring tools and their applications in computer science

Authoring tools have a significant role in authoring process, since their properties strongly affect the result. Well-designed authoring tools provide a proper basis for both design and implementation. This chapter gives background information of authoring tools and CAI courses on computer science. It consists of an introduction to authoring tool types and a short review on authoring tools used for preparing CAI courses on computer science. In addition, some CAI courses on computer science are discussed briefly.

2.1 CAI authoring tool types

CAI authoring tools can be divided into four categories: (1) high level programming languages, (2) authoring languages, (3) authoring systems, and (4) authoring shells (Young and Knezek, 1989). The main difference between these categories is the level of readiness of authoring, that is, how much effort an author must put into the implementing process of a CAI course. CAI courses created using these tools may vary from simple textual or graphical displays to very complex presentations that consist of graphics, pictures, and sound produced by the peripherals.

2.1.1 High level programming languages

This type includes the high level programming languages, which are the most flexible tools but which also require the most familiarity with programming. Production of teaching applications by these tools is slowest but the use of a high level programming language is justified if there exist some special requirements for the final application. High level

programming languages are especially useful when other authoring tools are inappropriate to meet the demands that an application requires. A complex simulator is a good example of a teaching program that would probably require a high level programming language.

When high level programming languages are used there is a risk that the quality of a teaching application suffers a loss since the technical implementation requires plenty of energy. Further, the experts of teaching subject domains, e.g., professors and lecturers, are usually not able to use tools of this type. In such cases the transfer of knowledge from experts to the designers may cause some loss of pedagogical quality. BASIC, C, Lisp, Logo, and Pascal are examples of high level programming languages.

2.1.2 Authoring languages

To create a teaching program by using an authoring language is not as demanding as using a high level programming language because there are predefined structure specific commands for developing teaching programs in authoring languages. For example, in an authoring language there might be a command 'QU' for a question to be asked a student, or a command 'R' for a response to be received. An advantage is that the usage of authoring languages reduces the time required for a teacher/designer to create a CAI course. In addition, authoring languages can be used for building the prototypes because these can be done quite fast. Afterwards, if the prototype is promising, the final application can be implemented by using a high level programming language to gain the fastest possible execution speed. Coursewriter, NATAL, Pilot, and Tutor can be mentioned as examples of authoring languages.

2.1.3 Authoring systems

An authoring system is usually a menu driven tool which eliminates or minimises the need for computer programming skills (Young and Knezek, 1989). Thus, those CAI course designers who are not experienced programmers are also able to produce technically appropriate applications by using these tools. Previously authoring systems mainly followed the behaviorist learning model when authors were only able to select the options the authoring system provided and organise the instructions in such order that the authoring system imposed. Modern authoring systems are more flexible than that. Nowadays it is possible to select pre-designed modules and, if necessary, to join with them some parts created by an authoring language. Some authoring systems include an authoring language of their own while some other authoring systems permit the usage of external authoring languages.

When using an authoring system the designer is able to guide a student in many ways. In some segments of an application the student must proceed linearly while in some other

segments he is provided branching opportunities. Branching gives the student liberty to select the topics more important to him and skip the less important ones.

It is usually possible to integrate several peripheral devices with the applications created by authoring systems. Thanks to these peripheral devices, which include video or laser disk players, speech synthesizers, and touch sensitive screens, the applications can be built more varying and thus more interesting. The touch sensitive screens are especially useful for increasing the interaction between a student and an application. Examples of this type of tools are PHOENIX, DECAL, IconTutor, InfoWindow, LS1, Course Builder, and Authorware.

2.1.4 Authoring shells

Authoring shells are the most restricted authoring tools but also the easiest ones to use. There is typically a pre-designed program skeleton into which the designer fills the instructional content. In a skeleton there are ready written routines, for example, for representing questions and analysing answers. The advantage of authoring shells is that the designer does not have to build everything from scratch. Further, the execution speed of these shells written in a high level language is very high. Also, the original code of an authoring shell can usually be changed thus it provides a designer with sufficient programming skills an opportunity to make improvements.

Authoring shells are available for diverse tasks. The simplest ones perform only one function, such as teaching vocabulary words or a single mathematicaloperation like adding or subtracting. More complex authoring shells permit to include several functions in the final application. In general, the more complex the authoring shell is the more difficult it is to modify, although the authoring shell is usually delivered containing the instructions for making modifications. Programming skills are useful though not necessary for modifying. One example of authoring shells is the Shell Games.

2.2 On CAI authoring tools used for courses on computer science

This section presents shortly some examples of the use of CAI authoring tools in computer science. Example courses are listed by authoring tool types in Table 2.1.

The dominating authoring tools in computer science applications seem to be high level programming languages despite the fact they were not originally developed for that kind of usage. This is a bit surprising since authoring with programming languages is time-consuming, even though there exist toolboxes containing useful modules. The reason may be that those familiar with high level programming languages prefer a tool they know in advance, rather than learning to use a new tool. In addition, with high level programming languages capable authors can create exactly what they want.

Hypertext and hypermedia tools, like high level programming languages, are not real CAI authoring tools as they are usually missing educational support. These tools are probably

used since it is possible to create impressive presentations fairly easily. Unfortunately, the educational support of such presentations remains minor. There is nothing wrong to use these tools for creating CAI materials as long as the author remembers to implement the educational properties.

CAI authoring languages and authoring systems are seldom used for creating courses on computer science. This is odd because there exist authoring systems with versatile properties that could be used fairly well. Authoring systems might be used more than they have been used so far, if they would offer the same power of expression as programming languages combined with flexibility, ease of use and the support for creating educational content.

Table 2.1: Some CAI applications on computer science by authoring tool types.

Authoring tool type	Name of the tool	Name of the application	Reference
High level programming languages	Actor	DBTool	Lim and Hunter, 1992
	Ada	Portable Diners	Feldman, 1992
	C	GATutor	Prince et al., 1994
		ISETL	Baxter et al., 1990
		Netsim	Barnett, 1993
		Trainset	Brown, 1993
		WATSON	O'Neal and Kurtz, 1995
	C++	DAPPLE	Kotz, 1995
	Cobol	APPGEN*	Osborne, 1992
	Common Lisp	FLAIR	Ingargiola et al., 1994
		HyperComments	Linn, 1992
		Portable AI Lab	Rosner and Baj, 1993
	Pascal	AAPT	Sanders and Gopal, 1991
		APPGEN*	Osborne, 1992
		ArcadeLaboratory*	Cagnat et al., 1990
		BALSA, GAIGS	Schweitzer, 1992
		KLYDE	Berque et al., 1994
	Prolog	Intelligent Fortran Adviser	Meehan et. al, 1991
		NLinterface	Lees and Cowie, 1996
	Sim++, ObjectStore	HASE	Coe et al., 1996
	Tcl/Tk	Gyacc	Lovato and Kleyn, 1995
Hypertext and hypermedia tools	Guide	CLEM	Boyle et al., 1994
	HyperCard	ArcadeLaboratory*	Cagnat et al., 1990
		Digital World	Fagin, 1994
		Gateway labs	Cowley et al., 1993
		Pascal Template Library	Linn, 1992
		Visual simulator	Barnett, 1995
	SuperCard	Lisp Template Library and Lisp Perspective Library	Linn, 1992

	Toolbook	UIDTutorial	Barrett, 1993
Authoring languages	NATAL	--	Gee and McArthur, 1991
Authoring systems	Authorware	MuPMoTT	Marsden and O'Connell, 1996
	AUTOOL	COSTOC courses	--

* Designed using several tools.

Authoring shells have not been used at all. The explanation might be that persons who author courses for computer science feel that authoring shells do not fulfill their needs and, since they are able to program, they prefer to use tools familiar to them, namely, programming languages.

2.3 Overview on computer-aided courses on computer science

This section introduces briefly some CAI applications designed for teaching computer science at the university level (see Table 2.2 and Table 2.3). Applications are divided into nine groups by the ACM (Association for Computing Machinery) computing review classification with slight modifications. This classification is useful, since it covers computer science domain very well and, consequently, it provides a good review of CAI applications. It must be remembered that this is only a small sample of all CAI applications and experiments done within computer science.

Table 2.2: Some CAI applications on computer science of the 1990's.

ACM Classification	Domain of the application	Name of the application	Reference
Introductory and survey	Introduction to computer science	ISETL	Baxter et al., 1990
		Gateway Laboratories	Cowley et al., 1993
			Baldwin, 1996
		WATSON	O'Neal and Kurtz, 1995
Computer systems organisation	Computer architecture	HASE	Coe et al., 1996
	Computer networking	VNET	Tymann, 1991
		Netsim	Barnett, 1993
		NetCp	Finkel and Chandra, 1994
Software	Introductory programming	ITEM/IP	Brusilovsky, 1992
		MULE	Barr and Smith King, 1995
		ANNET	Liffick and Aiken, 1996
		AAPT	Sanders and Gopal, 1991
	Fortran programming	--	Meehan et al., 1991
	Machine and assembly languages	Visual simulator	Barnett, 1995
	Modula-2 programming	CLEM	Boyle et al., 1994
	Parallel programming	Portable Diners	Feldman, 1992

		eText	Rifkin, 1994
		DAPPLE	Kotz, 1995
	Operating systems	SR language	Hartley, 1992
		NL interface	Lees and Cowie, 1996
Data	Data structures (using Ada)	--	Silver, 1991
	File structures	VDAM	Foster and Hughes, 1991

Table 2.3: Some CAI applications on computer science of the 1990's.

ACM Classification	Domain of the application	Name of the application	Reference
Theory of computation	Automata	FLAP	LoSacco and Rodger, 1993
		NPDA	Caugherty and Rodger, 1994
	Algorithms	Arcadelaboratory	Cagnat et al., 1990
		KLYDE	Berque et al., 1994
	Genetic algorithms	GATutor	Prince et al., 1994
	Grammars	Gyacc	Lovato and Kleyn, 1995
	Neural networks	Neuralis	Khuri and Williams, 1996
	Parsing	LLparse, LRparse	Blythe et al., 1994
Mathematics of computing	Mathematical concepts of computer science	Prologb	Neff, 1993
	Queueing theory (using computer algebra systems)	--	Fitzgerald and Place, 1995
Information systems	Databases	DBTool	Lim and Hunter, 1992
	Real-time and distributed computing	Trainset	Brown, 1993
	User interface design	UIDTutorial	Barrett, 1993
Computing methodologies	Artificial intelligence	Portable AI Lab	Rosner and Baj, 1993
		FLAIR	Ingargiola et al., 1994
	Expert systems	Automated Student Advisor	Harlan, 1994
Computing milieux	Technological literacy	Digital World	Fagin, 1994
	Systems analysis and design	APPGEN	Osborne, 1992
	Analysis techniques	MuPMoTT	Marsden and O'Connell, 1996

These sample applications indicate that almost all subdomains of computer science are applicable in a computer format. Most of the applications listed above are considered as educational tools, rather than traditional CAI courses. This means that those applications are mainly intended to provide supplementary material for traditional classroom teaching. It

would require a more detailed study to be able to decide the educational quality of these applications.

3 Human learning theories

Human learning theories

The human learning process has been investigated for centuries. Despite various studies, researchers have not been able to present a description for one universal learning process. Since there exist many learning styles and learning theories, it is difficult to decide which ones should be used in a CAI course design process. This chapter introduces human learning theories that have been applied to CAI courses on computer science. Four of the represented theories are discussed in more detail and a sample application of each is described. This chapter will be used as a background information for the evaluation of CAI courses on computer science.

3.1 Introduction to learning theories relevant to CAI

Dozens of learning theories have been presented in this century. These theories focus on diverse topics, such as human learning, animal learning, neuropsychology, and learning disabilities. Considering the topic of this study, the usage of computer-aided instruction (CAI) in teaching computer science, the human learning theories are the most interesting ones. These theories can be divided into three categories according to their relevancy to this study. The first category consists of theories that have been applied to CAI in some domain. The second category includes those human learning theories that have had a notable significance on the theories of the first category. Those theories that do not belong to either of the previous categories form the third category. From now on, the theories belonging to the first category are discussed. Theories belonging to the second category are mentioned only if they have had a significant impact on the development of the first group theories.

We examined up to 50 human learning theories and classified them according to their relevancy to CAI. Nineteen theories were found that have been applied to CAI. A closer

analysis revealed that these theories fall into three groups according to their relations to each other, namely historical theories, “chunking theories” and learner-centered theories. Three CAI theories were left outside of these three groups as they do not seem to have any relationship to the others. These three theories form the fourth group named separate theories.

3.1.1 Historical theories

The first group consists of two theories applied to CAI, namely *operant conditioning* (Skinner, 1938) and *mathematicallearning theory* (Atkinson, Bower and Crothers, 1965). The operant conditioning theory of Skinner (1938) has a notable role in the history of CAI because it was applied to instruction in a form of programmed instruction which can be considered as the first real form of CAI. Knezek (1988) has represented the evolution of CAI in which the teaching machine developed by Pressey in 1925 is considered as the first form of CAI. Throughout the 1920s and the early 1930s, Pressey experimented with several machines, but, as Sloffer (1996) puts it, their impact on educational technology at that time was negligible, although Pressey "remained confident that automated instruction would eventually generate an 'industrial revolution' in education" (Saettler, 1990). Skinner’s programmed instruction had, however, long-term effects on the evolution of educational technology. As Li (1996) points out, it had a strong influence on the development of the systems approach, which emphasises the process of instruction rather than media. Further, it formed the basis for individualised instruction and for the development of CAI.

Operant conditioning is based on behaviorism according to which learning means forming associations between stimuli and responses and can be explained without referring to any unobservable internal state. The nature and frequency of the stimulus-response (S-R) pairings determine the strength or weakness of associations. Operant conditioning differs from previous forms of behaviorism (e.g., Thorndike, 1913; Hull, 1940) in the sense that the organism can respond instead of only eliciting response due to an external stimulus.

Mathematical learning theory of Atkinson, Bower and Crothers (1965) has several predecessors among behaviorist theory, such as *drive reduction theory* (Hull, 1940) and *stimulus sampling theory* (Estes, 1950), which have tried to describe and explain behavior in quantitative terms. Atkinson et al. have been interested in optimising instruction, as the principles of his theory indicate (Kearsley, 1996):

1. It is possible to develop an optimal instructional strategy for a given individual provided that a detailed model of the learning process is available, and
2. Optimal learning performance can be achieved by giving each individual sufficient time to learn.

Mathematical learning theory has been applied to computer-aided language learning.

These first group theories represent behaviorism that has been found to be too simple to explain human learning. Therefore, CAI programs based on these theories are obsolescent.

3.1.2 "Chunking theories"

The name of this group, "chunking theories," is derived from the common idea of using *chunks* to represent information in all theories of this group. The idea of *chunking* is that short term memory can only hold five to nine chunks of information where a chunk is any meaningful unit, such as digit, word, chess position, or people's faces. These theories, which are closely related to each other, are a *General Problem Solver, GPS* (Newell and Simon, 1972), *GOMS* (Goals, Operators, Methods and Selection rules) (Card, Moran and Newell, 1983), *ACT** (Adaptive Control of Thought*, * belongs to the name) of Anderson (1983), and *Soar* (Laird, Newell and Rosenbloom, 1987). These four theories have a common background which is the *information processing theory* of Miller (1956). Miller has presented two ideas, that is, chunking and the capacity of short term memory and *TOTE* (Test-Operate-Test-Exit), which are essential to cognitive psychology and the information processing framework. TOTE was intended to replace the stimulus-response as the basic unit of behavior. The idea of a TOTE is that a goal is tested to see if it has been reached, otherwise an operation is performed to reach the goal; this until it is finally reached or abandoned.

General Problem Solver, GPS was an attempt to simulate human problem solving by means of a computer simulation program (Ernst and Newell, 1969; Newell and Simon, 1972). In GPS, cognitive models were specified by using *productions*, which were later used in other theories, such as ACT* (Anderson, 1983) and Soar (Laird et al., 1987). GPS was supposed to be a general problem-solver using the principle of breaking a problem down into subproblems and solving each of those. Unfortunately GPS failed. It was applicable only to "well-defined" problems, such as proving theorems in logic or geometry, word puzzles and chess.

GOMS model (Card, Moran and Newell, 1983) tries to explain those cognitive skills that are needed in human-computer tasks. The cognitive structure applied within GOMS has four components: (1) a set of goals, (2) a set of operators, (3) a set of methods for achieving the goals, and (4) a set of selection rules for choosing among competing methods. The interpretation of all cognitive activities is done in terms of searching a problem space, which is essential in GOMS as well as in Soar (Laird et al., 1987). GOMS has provided a basis for predicting the methods and operators users apply in the performance of computer-based tasks, such as using text editors and graphics systems (Olson and Olson, 1990). Furthermore, it has been used together with the *minimalist model* (Carroll, 1990) to develop documentation in a computer format (Gong and Elkerton, 1990).

The ACT* (Adaptive Control of Thought*) theory (Anderson, 1983) is based on the original *ACT theory* (Anderson, 1976) and the model of semantic memory called *HAM* (Anderson and Bower, 1973). Anderson (1990) has criticised his theory and provided (1993) the outline for a broader development of the theory. ACT* concentrates on memory

processes and introduces three memory structures: *declarative memory* that links propositions, images and sequences by association, *procedural memory* (also long-term memory) that represents information in the form of productions, and *working memory* that is the most highly activated part of long-term (procedural) memory. Anderson, Boyle, Farrell and Reiser (1987) have used ACT* as the basis for intelligent tutors.

Soar learning theory (Laird, Newell and Rosenbloom, 1987) has been strongly influenced by GPS and GOMS. Like both of them, Soar is built upon the idea that all cognitive acts are some form of search task in a problem space. It uses productions for expressing the human cognition and it has only one procedural memory. Soar, like Miller's information processing theory, applies chunking as the primary mechanism for learning. Soar has been used as the basis for systems that configure computer systems and formulate algorithms in a seemingly intelligent way.

Altogether "chunking theories" form a consistent research work. GPS was a very ambitious attempt but unsuccessful for the reason of the difficulties to implement human cognition in a computer format. Nevertheless, GPS, as well as information processing theory, has given a good framework for GOMS and Soar theories, which can be considered as continuations of the research started with GPS. Both GOMS and Soar have been applied to varying computer tasks and the research still continues.

3.1.3 Learner-centered theories

All nine learning theories in this group emphasise that teaching should be learner-centered. Learning theories of this group are *conditions of learning theory* (Gagné, 1965), *andragogy Theory* (Knowles, 1975), *criterion referenced instruction theory* (Mager, 1975), *functional context theory* (Sticht, 1975), *symbol systems theory* (Salomon, 1979), *cognitive flexibility theory* (Spiro, Coulson, Feltovich and Anderson, 1988), *anchored instruction theory* (Bransford and Cognition & Technology Group at Vanderbilt, 1990), *minimalism theory* (Carroll, 1990), and *situated learning theory* (Lave and Wenger, 1991). Short descriptions of each theory are presented.

The main idea of conditions of learning theory is that there exist different types or levels of learning and each of them requires different types of instruction. Further, each type of learning requires different internal and external conditions. Gagné and Briggs presented in 1974 a set of instructional events and corresponding cognitive processes which should provide the necessary conditions for learning and serve as the basis for designing instruction and selecting appropriate media. Gagné's learning theory has been applied to the design of instruction in several domains (Gagné and Driscoll, 1988), though the original formulation, which the theory is based on, was applied to computer-based military training programs (Gagné, 1962).

Andragogy theory (Knowles, 1975) is an adult learning theory that pays attention to the fact that adults are self-directed and expect to take responsibility for decisions. Andragogy has been affected by experiential learning theory (Rogers, 1969) and, according to Knowles (1984), it is applicable to the design of personal computer training.

Another theory that resembles experiential learning and andragogy is criterion referenced instruction theory, CRI (Mager, 1975). It also shares some ideas of Gagné's conditions of learning theory, such as task hierarchies and objectives. CRI is meant for the design and delivery of training programs involving a variety of different media, for example, videotapes and computer-based instruction.

The functional context theory (Sticht, 1975) focuses on taking into account the experience of learners and their work context when designing the learning environment for them. The main idea is that learning of new information is easier if the learner is able to take advantage of his knowledge and skills to assimilate the new information. Functional context theory has been applied, for example, in a form of computer-based instruction in the U.S. Navy. It is closely related to situated learning theory (Lave and Wenger, 1991) regarding the emphasis on context during learning.

Salomon's symbol systems theory (1979) focuses on investigating the effects of media on learning. Salomon points out that symbol systems of media affect the knowledge acquisition in many ways, such as highlighting different aspects of content, varying with respect to ease of recoding, and differing with respect to the kinds of mental processes they call on for recoding and elaboration. Symbol systems theory is similar to the *aptitude-treatment interaction theory* (Cronbach and Snow, 1977). Salomon's research has concentrated on film and television, but recent work is on computers (e.g., Salomon, Perkins and Globerson, 1991).

The nature of learning in complex and ill-structured domains is the object of research in cognitive flexibility theory (Spiro et al., 1988). Special interest is on the transfer of knowledge and skills. Therefore, the research is focused on presenting information from different perspectives and using several case studies that present various examples. Cognitive flexibility theory has been influenced by other constructivist theories, such as *genetic epistemology* (Piaget, 1936), *subsumption theory* (Ausubel, 1963) and *constructivist theory* (Bruner, 1966), and it is related to symbol systems (Salomon, 1979). The scope of this theory is to support the use of interactive technology.

Minimalism theory (Carroll, 1990) has its roots in four theories: genetic epistemology of Piaget (1936), constructivist theory of Bruner (1966), experiential learning theory of Rogers (1969), and andragogy theory of Knowles (1975). Carroll intended his theory especially for the design of training materials for adult computer users. In minimalism, like in experiential learning and andragogy, the learner's experience plays an important role in learning.

Anchored instruction theory of Bransford and Cognition & Technology Group at Vanderbilt, CTGV (1990) is another theory that focuses on technology-based learning. It was aimed at the development of interactive videodisk tools that encouraged students and teachers to pose and solve complex, realistic problems. The concept of *anchor* is derived from the function of video materials, which form the basis for all later learning and instruction. Anchored instruction theory shares its emphasis on the use of technology-based learning with cognitive flexibility theory (Spiro et al., 1988).

Situated learning theory (Lave and Wenger, 1991) emphasises that learning is a function of the activity, context and culture in which it occurs. It also stresses the importance of social interaction. Situated learning has been influenced by Gibson's *information pickup theory* (1966) and Vygotsky's *social development theory* (1962). Cognition & Technology Group at Vanderbilt (1993) has applied this theory in the context of technology-based learning activities for schools that focus on problem-solving skills.

Most theories within this group emphasise experience, motivation, social interaction and intelligence, and are focused on adult learning. Thus, the applications based on these theories are mostly computer-based training programs or similar ones.

3.1.4 Separate learning theories

Three learning theories have been applied to CAI but we do not include them in any of the three groups are discussed next. These three theories are considered as separate theories since they have no clear connection to the other theories. The three theories are the *script theory* of Schank and Abelson (1977), *repair theory* of Brown and VanLehn (1980), and *component design theory* of Merrill (1994).

The main interest in script theory (Schank and Abelson, 1977) has been the structure of knowledge. It is based on *conceptual dependency theory* (Schank, 1975) which focused on the representation of meaning in sentences. Central concepts of script theory are *scripts, plans* and *themes*, which are for handling story-level understanding. Script theory is aimed at explaining language processing and higher thinking skills. It has been applied, for example, to the development of intelligent tutors (Schank, 1991).

Within the framework of repair theory, Brown and VanLehn (1980) have examined how people learn procedural skills. Their special interest has been how and why people make mistakes. They propose that when an individual is heading for an impasse, i.e., a procedure cannot be performed, he applies various strategies, or scripts, to overcome it. Repair theory is based on the study of children solving arithmetic problems and it is implemented as a computer model called Sierra (Kearsley, 1996).

The component design theory (Merrill, 1994) is strongly based on Merrill's previous theory called *component display theory* (Merrill, 1983). Component design theory emphasises the course structure and instructional transactions rather than presentation forms,

as Kearsley (1996) points out. Furthermore, the learner control strategies introduced in component display theory are replaced by the advisor strategies in component design theory. The old CDT theory has been used in the design of the lessons in the *TICCIT* (Timeshared Interactive Computer-Controlled Information Television) computer-based learning system (Merrill, 1980) and as the basis for the *Instructional Quality Profile*, which is a quality control tool for instructional materials (Merrill, Riegeluth and Faust, 1979). The development of the new CDT theory has been closely related to work on expert systems and authoring tools for instructional design (e.g., Li and Merrill, 1991; Merrill, Li and Jones, 1991).

Theories gathered into this group have no special relation to each other. Instead, they have typically been a result of the research work of one group over a long period, for example Merrill has developed his CDT theory since the first half of 1980's.

3.1.5 Summary of the theories

The learning theories behind CAI have been discussed shortly above. The goals of instruction and typical features of the presented theories are collected in Table 3.1.

The review shows that the assortment of theories is very wide. Some of these learning theories try to explain adult learning and, therefore, emphasise the importance of experience and existing knowledge. Some other theories stress the social interaction as the essential part of learning process. All these aspects are important for learning but it should be remembered that people are individuals and learn in different ways. Therefore, what suits one person may not be convenient for another.

Generally CAI theories concentrate on a certain age group (e.g., adults), a certain topic (e.g., elementary reading, language processing), certain skills (e.g., mathematical skills, higher thinking skills, problem solving skills), or a certain technology (e.g., film and television, interactive technology, computer-based training materials). This indicates that the complexity of human learning is a well-known problem and, thus, researchers do not try to create general learning theories that would explain human learning in all ages and all domains.

There exist some features that are common to most of the theories. These features include the motivation of the learner, which is emphasised almost in every theory, and the idea that learning is based on students' existing knowledge and happens through adding, changing or ruling out that knowledge. Motivation is necessary because if learners are not well-motivated, learning may remain superficial, for example, students know things by heart only, and real learning may not happen. In addition, it is common that knowledge and skills are learned different ways and the age of the learner affects to the learning process.

Table 3.1: The goals of instruction and typical features of some human learning theories.

Category	Name of the theory	The main goal of instruction	Typical features
Historical theories	Mathematical learning theory	Language learning in the context of computer based instruction	Quantitative terms for describing and explaining behavior
	Operant conditioning	Changes in behavior (stimulus-response)	Programmed instruction and teaching machines
"Chunking theories"	ACT*	Memory effects and higher order skills (e.g., geometry proofs and programming)	Productions, three memories (declarative, procedural and working)
	GOMS	Cognitive skills involved in human-computer tasks (e.g., text editing)	Problem space, set of goals, operators, methods, and selection rules
	GPS (General Problem Solver)	Problem solving using productions for specifying cognitive models	Production rules, problem breaking into subcomponents
	Soar	Architecture for human cognition (e.g., verbal learning tasks, reasoning)	Problem space, procedural memory and productions
Learner-centered theories	Anchored instruction	Problem solving (e.g., elementary reading, mathematical skills)	Technology-based learning, interactive videodisk tools using anchors
	Andragogy	Problem-solving on topics of immediate value (e.g., management development)	Adult learning: self-directed students take responsibility for decisions
	Cognitive flexibility	Transfer of knowledge and skills using different perspectives	Complex and ill-structured domains; use of interactive technology
	Conditions of learning	Intellectual skills	Different instruction for different learning outcomes on specific internal and external conditions of learning
	Criterion referenced instruction	Mastery of learning objectives (e.g., technical training including troubleshooting)	Training programs using a variety of media and a set of methods
	Functional context	Basic skills (e.g., reading), adult technical and literacy training	Experience of learners
	Minimalism	Skills of computer users (e.g., word processing, programming)	Adult learning, minimisation of passive forms of training
	Situated learning	Knowledge acquisition (e.g., problem solving skills)	Learning is a function of an activity, context and culture.
	Symbol systems	Knowledge acquisition	Effects of media on learning
Separate	Component	Cognitive skills	Two dimensions of learning with four

learning theories	design theory		primary and five secondary presentation forms
	Repair theory	Procedural skills	Repairs for impasses
	Script theory	Cognitive skills (e.g., language processing, higher thinking skills)	Scripts, plans and themes to handle story-levelunderstanding

3.2 Four learning theories of current interest

In this section, four learning theories of current interest and some example applications based on them are discussed more widely. We try to find at least one learning theory that could be used as a basis for our instructional criteria and these four theories seem to be the most promising candidates for our purpose.

These four theories represent thinking that considers teaching as a transformation of knowledge, not as a transmission of it. The selected four theories are conditions of learning theory, ACT*, cognitive flexibility theory, and situated learning theory, and the applications based on them are *GAIDA, Geometry Tutor, hypertext applications,* and *the Adventures of Jasper Woodbury,* respectively. A computer plays different roles within these theories. It is used for presenting information in conditions of learning, in cognitive flexibility theory and in situated learning theory, while ACT* tries to model human intelligence using a computer.

3.2.1 Conditions of learning theory and GAIDA

Conditions of learning theory (Gagné, 1965) emphasises that successful learning requires certain internal and external conditions. Further, there exist different types of learning which require different types of instruction. These types of learning, or outcomes that learning is expected to have, as Gagné and Briggs (1974) put it, should be the starting-point for designing of instruction. The categories of learning outcomes are:

- intellectualskills,
- cognitivestrategies,
- verbalinformation,
- attitudes, and
- motor skills.

Intellectual skills involve 'knowing how,' rather than 'knowing that.' Intellectual skills make the human individual competent and enable him to respond to conceptualisations of his environment. Further, the learning tasks involved with intellectualskills can be divided into hierarchicalcategoriesaccordingtocomplexity:

- stimulus recognition,
- response generation,
- procedure following,
- use of terminology,
- discriminations,

- concept formation,
- rule application, and
- problem solving.

The importance of this hierarchy is that it helps to recognise prerequisites that should be completed to facilitate learning at each level. These prerequisites are found using a task analysis of a learning task. The hierarchy also serves as a basis for the sequencing of instruction.

Cognitive strategies are internally organised skills which govern learners' own behavior. They are used by learners to manage the processes of attending, learning, remembering, and thinking. These processes, for their part, activate and modify other learning processes.

Verbal information consists of three kinds of learning situations, namely, learning of labels or names, learning of isolated or single facts, and learning of organised information or knowledge. Verbal information serves as a prerequisite for future learning.

Attitudes are divided into two classes which are, firstly, the attitudes that the student has toward the school institution and things related to it, and secondly the attitudes that the school aims to establish or change as the result of instruction. As attitudes function to affect "approaching" or "avoiding," as Gagné and Briggs (1974) put it, they influence a large set of specific behaviors of the individual.

Motor skills make possible individuals' precise, smooth, and accurately timed execution of performances involving the use of muscles. A total performance can usually be divided into part-skills that occur simultaneously or in a temporal order. In order to learn this total performance, the integration of its parts, as well as the component part-skills themselves, must be learned.

GAIDA

GAIDA (Guided Approach to Instructional Design Advising) is an on-line, case-based system giving guidance for designing interactive courseware. It is strongly based on another project called *AIDA* (Advanced Instructional Design Associate), which is aimed at providing powerful on-line courseware authoring assistance. AIDA system (Armstrong Laboratory, 1996a) consists of lesson templates that guide the subject matter expert in developing courseware, reusable instructional strategy templates, integrated text, graphics, and video editors, an integrated simulation authoring capability, and an automatic test question generator. The templates are pre-programmed with expert knowledge of instructional strategies that are effective for such lesson objectives as identifying parts and functions of devices, executing operating and maintenance procedures, and interpreting various diagnostic procedures and information.

GAIDA (Armstrong Laboratory, 1996b) is both an authoring advisor and an interactive courseware collection, since it has two modes called guidance and lesson. These modes are

used side by side, since the user is provided with an explanation of using the nine instructional events of Gagné effectively to create interactive courseware, and this explanation is tied to the GAIDA database of cases allowing the user to jump from guidance mode to lesson mode as desired. The lesson database of cases contains computer-based instruction for a range of learning objectives and presentation techniques, including various multimediaapplications.

The target user group of GAIDA is novice courseware developers for whom it provides automated instructional design guidance. According to Armstrong Laboratory (1996b), GAIDA elaborates the nine events of instruction in the context of specific instructional goals such as identification, classification, checklist procedure, and memory procedure. These events of instruction are gain attention, indicate goal, recall prior knowledge, present material, provide learning guidance, elicit performance, provide informative feedback, assess performance, and enhance retention and transfer. In addition, GAIDA presents specific advice concerning the effective use of multimedia.

3.2.2 ACT* and the Geometry Tutor

Anderson has attempted to relate theoretical principles to acquisition of proceduralised skills in his ACT* theory (1983, 1987). For modeling cognitive skills he has used hierarchically arranged productions which he regards as the key element of skill acquisition. The problem solving is organised using both cognitive units formed by productions and the hierarchical goal structure specified by productions. Current active knowledge is referred as the working memory, in which a new, externally set goal gets active. In production memory, if there exists an established production for solving the given type of problem, then that production will be generated for the problem. In a case that there does not exist a known solution, a search for a solution must begin. According to Polson (1993), it is assumed that to solve novel problems people apply weak-problem solving to the declarative memory that they have about this domain. Newell and Simon (1972) have stated that this weak-problem solving solution comprises analogy, means-ends analysis, working backward, hill climbing, and forward search.

These general weak-problem solving rules are used for generating a solution to the problem. The existing declarative knowledge about the domain determines which weak method will be applied. A successfully solved problem produces a route consisting of the hierarchically organised productions. The route is found by a search through declarative memory for the necessary conditions. The next step is to transform this route to some efficient domain specific productions by compilation. The first phase of compilation is proceduralisation in which the route of productions is generated in the production memory. This relieves working memory since there is no longer any need to search declarative memory for the conditions that can be removed from the working memory. On the second

phase, the productions belonging to the route are compiled into a production that carries through the same task. The advantage of compilation is that it speeds up solving the future problems of this nature.

Geometry tutor

Geometry Tutor is an intelligent tutor for geometry problem solving instructing traditional Euclidean proof skills. According to Anderson (1993), geometry expertise is managing a difficult search task by means of heuristics. The goal of the Geometry Tutor is to teach students approximately 100 rules of inference that they can use for proving various geometry problems. It is important that students learn to recognise which rules are plausible ones among all applicable rules.

The intelligence of the Geometry Tutor is based on an expert model that knows when to apply various rules of inference. There exist various contextually bounded rules that recognise the situation in which a certain geometry inference is likely to be useful. In addition, there exist backward-inference rules for setting particular subgoals when it is considered reasonable. In order to follow students' moves there exist low-rated versions of legal rules of inference, which are much weaker and, therefore, they are not preferred if any higher rated rules are applicable. Each production rule has a rating that is a one-dimensional quantity associated with that rule. This rating should be considered as a mixture of cost and probability of success and it indicates how likely a production is to fire. Anderson (1993) states that this expert model is able to find proofs quite efficiently and it represents the kind of expertise that was building up in successful students as a function of their experience with different problems.

The user interface of the Geometry Tutor consists of three parts: the statement the student is trying to prove, the givens of the problem, and the problem diagram. The student uses a mouse to select a set of statements and enters a rule of geometric inference that takes these statements as premises. After that the student is asked to type in the conclusion that follows from the rule. For entering the conclusion the student is provided with a menu, which contains the relations and the symbols of geometry. Thus, one step of inference consists of the sequence of premises, rule of inference, and conclusion. The system updates the screen after each step. At the end the tutor shows a completed proof which consists of a graph structure connecting givens to the to-be-proven statement. This completed proof graph is enlightening in two ways, namely, students see how inferences combine to yield a proof, and the search inherent in proof generation is explicitly represented.

If the student has many difficulties with the proof the tutor points him to the key step in solving the problem. In addition, the tutor gives advice what to do and, if needed, shows how the specific conclusion can be proved. The student is able to use both reasoning forward from the givens and reasoning backward from the conclusions.

According to Anderson (1993), the proof graph makes concrete two abstract features of problem solving in geometry, namely, the logical relationships among the premises and conclusions, and the search process by which one hunts for a correct proof. He states that students have a great deal of difficulty with both of those constructs. The proof graph facilitates instruction on these abstract concepts by creating an external referent.

3.2.3 Cognitive flexibility theory and hypertext applications

Cognitive flexibility theory (Spiro, Coulson, Feltovich and Anderson, 1988; Spiro, Vispoel, Schmitz, Samarapungavan and Boerger, 1987) is a general theoretical orientation to knowledge acquisition and application in complex and ill-structured content domains. It is based upon cognitive learning theory and serves as a conceptual model for designing learning environments. According to Jonassen (1992), the major principles of cognitive flexibility theory include the following:

- avoids oversimplifying instruction,
- provides multiple representations of contents,
- emphasises case-based instruction,
- context dependent knowledge,
- knowledge construction, not transmission, and
- supports complexity.

Instructional materials should avoid oversimplifying the content domain. This principle is based on that if students are provided with oversimplified tasks they later on fail to understand their field at a more advanced level. Therefore, instruction should reflect the complexity that normally faces the practitioners.

Multiple perspectives or interpretations of the instructional content are represented when cognitive flexibility theory is applied. The reason for this is that in order to understand the complexity of the world individuals must perceive and reconcile its different interpretations. According to Gick and Holyoak (1983), the instructional use of multiple analogies facilitate the understanding of multiple mental representations. This understanding is needed for the transfer of acquired knowledge to novel situations, that is, in problem solving.

The advantage of the case-based instruction is that the more varied cases are used to illustrate the content domain, the broader the conceptual bases that they are likely to support, as Jonassen (1992) puts it. Multiple perspectives or themes that are inherent in many cases are the best ways to illustrate the ill-structuredness of any knowledge domain.

It is important to present knowledge in a situation that is representative of its normal use since, according to Brown, Collins and Duguid (1989), knowledge is best acquired in relevant situations that are likely to be encountered by the student as a practitioner. This is based on the notion that the same entity or objects may appear or function quite differently in different contexts (Jonassen, 1992).

Cognitive flexibility theory emphasises that students should be responsible for constructing their own knowledge representations in order to adapt and use it in novel situations. The reason for this is that when an individual integrates knowledge into his own knowledge structure it becomes more transferable to different problem domains.

Complexity of the instructed knowledge is supported since it is important that students recognise the inconsistencies in that knowledge and acquire it in a non-compartmentalised form. A method for supporting complexity is to present multiple representations of the same information and different thematic perspectives on the information. Students construct useful knowledge structures by comparing and contrasting the similarities and differences between these cases.

Hypertext applications

The usage of hypertext as an instructional application has been studied for several years. Jonassen (1992) argues that cognitive flexibility theory implemented in hypertext provides an effective model for designing and developing computer-based instruction to support advanced knowledge acquisition which is required by professionals to solve real-world problems. Jonassen (1991) has presented reasons that make hypertext a powerful environment in which to design, develop, and deliver most computer-based learning. The first of the reasons is the flexibility of hypertext which is based on its associative structure. Another reason is that hypertext and instructional systems share the same theoretical framework. In addition, a variety of instructional designs can be mapped directly onto hypertext, that is, it can well be used both as a design and development environment. Lastly, the differences between designers and users become less obvious, as hypertext systems become more collaborative.

Jonassen (1992) also states that for conveying or delivering traditional instruction hypertext is an inappropriate environment. He emphasises that students can learn from hypertext only if they have a useful, relevant, and constructive purpose for accessing information in the hypertext.

Hypertexts based on cognitive flexibility theory are discussed, for example, in Jacobson (1990), Spiro and Jehng (1990) and Jonassen (1992). One hypertext is presenting an overview of transfusion medicine by describing problems in blood transfusion from multiple perspectives. This knowledge domain is very wide and complex since it includes all disciplines in medicine which are involved with handling blood, and it entails both several basic sciences and clinical areas. According to Jonassen (1992), the hypertext concentrates on assessing risks, differential diagnosis, and prescriptive treatment of the range of transfusion problems to donors and recipients. Since transfusion medicine has grown more complex, additional learning has become necessary. Furthermore, the area of blood banking and hemotherapy has been incomplete and plagued with misinformation and opinions. These

are the reasons why the transfusion medicine hypertext, which is aimed to enhance the education of medical students, residents and practicing physicians, has been developed.

The transfusion hypertext operations are based on a database whose declarative knowledge is generated and structured using matrix analysis (Jonassen, Hannum and Tessmer, 1989). This database contains information of various kinds of transfusion events, causes of which may produce several possible symptoms or require several optional medical interventions. These transfusion events interact with the purpose of the transfusion, the physiology of the patient and many other factors that causes the ill-structuredness of indicators. As a result, no single case is able to represent all of the possible transfusion effects.

In addition to the knowledge base, other parts of the transfusion hypertext include a glossary, seven practice cases for training, three test cases, a set of 24 mini-cases related to the practice cases, and a library of handlers. The seven practice cases, also known as primary practice cases, are employed for orientating the program. Students start practicing using these cases and through them they have access to any needed information or actions, such as a transfusion medicine textbook, query important operatives in the case, order tests, or compare the current case to similar cases. Thus, students are provided with several points of views. In addition, the program presents analytic feedback specific to the student's selection when an action is taken.

Jonassen (1992) has evaluated flexibility theory to this transfusion hypertext. He reports that transfusion medicine is a complex knowledge domain which must be made clear to students by the instruction. Therefore, the tenet of flexibility theory stating that instruction should not be oversimplified supports the usage of the hypertext format. Similarly, hypertext format provides multiple representations of content which is another tenet of flexibility theory. Students using the transfusion hypertext may consult several people, such as the patient, resident, or attending physician, or they can seek information from a textbook. All these sources give students information from different perspectives. Further, the transfusion hypertext containing the seven cases and 24 related cases support case-based instruction, which is emphasised in cognitive flexibility theory. Furthermore, the cases provided in this hypertext are real transfusion cases, which is very important since students acquire knowledge best in relevant situations that are likely to be encountered by them as practitioners (Brown, Collins and Duguid, 1989). One of the tenets of flexibility theory states that learners must construct meaningful knowledge representations in order to adapt and use the information in novel situations. This demand is fulfilled in transfusion hypertext since students have to access and associate information in relevant, practical situations. Students have to build complex schemata consisting of procedural data in order to solve cases. Lastly, the transfusion hypertext supports both complexity of the domain by offering multiple perspectives on each case and related cases for students to compare and contrast

them. Similarly, the hypertext supports students acquiring non-compartmentalised knowledge.

3.2.4 Situated learning and the Adventures of Jasper Woodbury

Situated learning theory (Lave, 1988; Lave and Wenger, 1991) emphasises two principles. The first one states that knowledge needs to be presented and learned in an authentic context, that is, settings and applications that would normally involve that knowledge. The second principle says that learning requires social interaction and collaboration. Especially the first principle contrasts with the traditional classroom learning activities that involve knowledge which is often presented in an abstract form and out of context. Lave and Wenger state that learning is situated both in space, time and relation to social practice and, thus, learning is legitimate peripheral participation in a community of practice. This community of practice, which is a broad characterisation comprising all social relations, represents, to quote Heaney (1996), "a negotiated set of relations among persons, their actions, and the world over time and in relation to other tangential and overlapping communities of practice." The complex identity of each person consists of relationships within multiple communities of practice. Individuals' possibilities for learning are defined and limited by the structures of communities in which each individual participates. These structures also set terms for individuals' legitimate participation. Therefore, learning is an ongoing negotiation with communities of practice.

Further, learning is participation which means absorbing and being absorbed in the culture of practice. It can be centripetal or centrifugal, i.e., moving an individual inward toward more intensive participation, which is empowering, or moving an individual outward, keeping him on the periphery and preventing him from participating more fully, which is disempowering, respectively.

The theory of situated learning has been developed further by other researchers. As an example, Brown, Collins and Duguid (1989) emphasise the idea of cognitive apprenticeship:

> Cognitive apprenticeship supports learning in a domain by enabling students to acquire, develop and use cognitive tools in authentic domain activity. Learning, both outside and inside school, advances through collaborative social interaction and the social construction of knowledge.

The Adventures of Jasper Woodbury

The Adventures of Jasper Woodbury is a program focused on mathematical problem solving with links to science, history, social studies and literature. Its theoretical basis is the theory of anchored instruction (Bransford and CTGV, 1990) which is a specialised form of the situated learning theory. The idea of the anchored instruction is to anchor instruction in meaningful problem solving environments that allow teachers to simulate in the classroom some of the advantages of "in-context" apprenticeship training (Brown et al., 1989).

According to CTGV (1994), the special emphasis is on contexts that help novices move from a general understanding of a complex problem to one in which they learn to generate and define the distinct subgoals necessary to achieve an overall goal. The creation of these instructional materials that produce generative learning activities has been guided by seven basic design principles:

- video-based format (teacher and budget friendly),
- narrative with realistic problems (rather than a lecture on video),
- generative format (students must generate the subproblems to be solved at the end of each story),
- embedded data design (all the data needed to solve the problems are in the video),
- problem complexity (each adventure involves a problem of at least 14 steps),
- pairs of related adventures (discussions of similarities and differences help students focus on general characteristics), and
- links across the curriculum (to supplement the standard mathematics curricula in a way that motivates students to explore mathematics in more detail).

The Adventures of Jasper Woodbury are video-based series for learning problem posing, problem solving, reasoning, and effective communication. At the end of each 15-20 minutes adventure, the major character or group of characters of the movie have a problem to be solved. The students in the classroom must solve that challenge before they are allowed to see how the movie characters solved it.

The adventures of Jasper Woodbury, like the other applications of anchored instruction created by CTGV, has several goals, such as, highlight the uses of knowledge, help students develop representations of their experiences that set the stage for positive transfer, promote collaborative learning, and help students improve their abilities to accomplish goals that are more holistic than typical tests of student achievement. These goals are fulfilled using various methods. Anchors be used to help students see the need for new learning and set important learning goals. According the CTGV (1993), it appears that the experience of identifying learning goals and then setting out to accomplish them is a very important aspect of adaptation in everyday life. Secondly, students learn to represent multiple solution paths, to summarise data, and to discuss characteristics of various types when using the Jasper series. Furthermore, collaboration is necessary since the problems depicted in anchors are complex and, thus, any individual student is unlikely able to solve them completely. Lastly, the holistic goals, such as, beginning with a general indication of a problem, generating the subgoals to solve it, and then doing so, are more important than producing higher scores on student achievement tests. In addition, students are encouraged to present their ideas and arguments to others and criticise others arguments.

The CTGV (1993) reports their experiences during the anchored instruction research project. For example, it is a big challenge to change the culture in classrooms, that is, the change of the role of teachers from tellers to coaches and fellow learners. Further, it is better to start with simple, familiar tools, such as a videodisk player and introduce more

sophisticated computer technologies later. In addition, they stress the importance of the school-wide and community-wide support for new projects. For this reason they have attempted to provide additional support by creating some key assessment instruments that would allow teachers to show others what the students have learned.

4 Criteria for evaluation of CAI courses in computer science

Criteria for evaluation of CAI courses in computer science

Several points of view must be considered when designing and evaluating a CAI course, such as domain-based demands, instructional demands, user interface demands, and pragmatic demands. These requirements are discussed in a case of computer science and a set of criteria is elicited. Since there exist factors that are related to several demands, such as interactivity and feedback, which both are related to instructional demands and user interface demands, they are discussed in both sections.

These evaluation criteria can be considered as the embodiments of different evaluation perspectives to CAI courses (see Figure 4.1). These perspectives concentrate on different properties of a CAI course, thus, together they form a consistent view of the examined course.

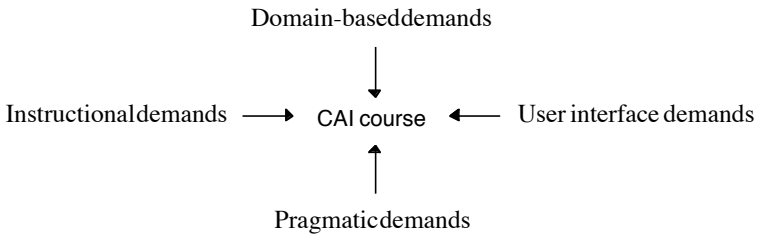

Figure 4.1: Evaluation perspectives to a CAI course.

The criteria can be modified for designing and evaluating CAI courses in other domains by defining the criteria based on the desired domain and replacing the criteria based on computer science with them.

4.1 Background for the selection of the criteria

Domain-based criteria

When defining the domain-based criteria, it is necessary to know the domain properly. In addition, it is important that the concepts and methods of that domain are generally approved within the domain, otherwise the criteria will not be applicable throughout the domain. If the domain is disputed, it is better to concentrate on subdomains and state clearly the limits of it, rather than try to create criteria on the whole domain.

Our criteria for the domain of computer science are based on the report of Denning et al. (1989) which has been condensed from the Report of the ACM Task Force on the Core of Computer Science. This report presents a framework for the discipline of computing and a basis for computing curricula. The task-force was given three general charges (Denning et al., 1989):

1. Present a description of computer science that emphasises fundamental questions and significant accomplishments. The definition should recognise that the field is constantly changing and that what is said is merely a snap-shot of an ongoing process of growth.
2. Propose a teaching paradigm for computer science that conforms to traditional scientific standards, emphasises the development of competence in the field, and harmoniously integrates theory, experimentation, and design.
3. Give a detailed example of an introductory course sequence in computer science based on the curriculum model and the disciplinary description.

The paradigms stated for the discipline defined by the task-force provide the context for fulfilling the charges described above. The task-force ended up to define computing as a discipline and its division into subareas. These paradigms, the definition, and the subareas are discussed in detail in section 4.2. The task-force emphasised that their description of computing is “living," that is, it can be revised from time to time to reflect maturity and change in the field.

Like in other domains, in computer science the goal of education is to *develop competence* in the discipline. According to Denning et al. (1989), the educational process that leads to competence has five steps: (1) motivate the domain; (2) demonstrate what can be accomplished in the domain; (3) expose the distinctions of the domain; (4) ground the distinctions in history; and (5) practice the distinctions. Computer science has two broad areas of competence, namely, *discipline-oriented thinking* and *tool use*. The first, which Denning et al. (1989) consider the primary goal for computing majors, is the ability to invent new distinctions in the field, leading to new modes of action and new tools that make those distinctions available for others to use. The latter is the ability to use the tools of the field for effective action in other domains.

Instructional criteria

The instructional criteria are based on instructional events of conditions of learning theory (Gagné, 1965). Even though this theory does not represent the latest trends emphasised in human learning (e.g., constructivism), it offers an appropriate basis for our evaluation criteria. The reason is that the emphasis of this theory, that is, there exist different types of learning (see Section 3.2.1), supports well our subject domain. By this we mean that teaching computer science requires instruction on all learning types (i.e., intellectual skills, cognitive strategies, verbal information, attitudes and motor skills) emphasised in conditions of learning theory and the instructional events offer a proper framework for the criteria. It is possible that our instructional criteria are not applicable to all domains but in our case they work fine.

In Table 3.1 there are presented the goals of instruction and typical features of learning theories applied to CAI. On comparison of the theories, we noticed that there does not exist many theories that could have been used as the basis for our criteria. The historical theories are out of question since they are obsolete. The "chunking theories" are promising but we did not become convinced of their functionality.

The learner-centered theories form the most promising group of theories. However, most of the theories of this group proved to be inappropriate for our needs. For example, some theories emphasise the learning situation and the context of learning (e.g., anchored instruction and situated learning) which are less important within computer science. Some other theories stress learners' experience (e.g., andragogy, functional context, and minimalism) which is important but cannot be expected from young adults that computer science students mostly are. We considered the conditions of learning theory the most suitable learning theory for our purpose since it meets all our needs. Other theories that might have been useful are criterion referenced instruction, which shares some ideas with the conditions of learning theory (i.e., learning needs and outcomes), and cognitive flexibility theory that could be applicable to CAI courses on computer science in advanced level..

The separate theories seem too unilateral for our purposes. Therefore they were disregarded.

4.2 Criteria based on subject

Computer science and computer engineering, commonly called computing, are domains which aim at solving problems using computers. Denning et al. (1989) have stated that these two disciplines have three paradigms that are partly common to both of them. These paradigms are theory, abstraction and design. Theory, which is based on mathematics, consists of four steps:

1. characterise objects of study (definition);
2. hypothesise possible relationships among them (theorem);

3. determine whether the relationships are true (proof); and
4. interpret results.

Abstraction, which is based on the experimental scientific method, also consists of four steps used for investigating a phenomenon:

1. form a hypothesis;
2. construct a model and make a prediction;
3. design an experiment and collect data; and
4. analyse results.

Design, which is based on engineering, consists of four steps that are used for constructing a system to solve a given problem:

1. staterequirements;
2. statespecifications;
3. design and implement the system; and
4. test the system.

Computer science focuses on theory and abstraction, whereas computer engineering focuses on abstraction and design, as Denning et al. (1989) have stated. From now on, computer science and computer engineering will be combined under the name of computer science.

Different methods play an important role in computer science instruction, since students are taught to recognise problems, find solutions to these problems, and finally, implement these solutions in a computer format. Good knowledge of both mathematics and logic are necessary for computer scientists since implementing complex computer systems requires ability both to understand abstract problems and to draw conclusions. To quote Denning et al. (1989):

> The discipline of computing is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, "What can be (efficiently) automated?"

At the university level, computer science education emphasises general abilities to solve computer tasks and good general education on available computer tools and techniques, rather than practical skills. Therefore, instruction of computer science mostly consists of teaching abstract concepts and methods, though practical skills cannot be totally forgotten.

Computer science is a fast developing discipline, which causes great challenges to educators. Changes both in hardware and software force frequent assesment of the computer science curriculum. This means that the curriculum should be updated to cover new concepts and methods. These changes reflect also to CAI courses on computer science that should be flexible to update. Since changes within the discipline happen fast, updating is necessary in order not to waste lots of design and implementation effort put into CAI courses on this domain.

The domain of computer science contains a wide range of diverse topics. It can be divided roughly into the following subdomains (Denning et al., 1989):

- algorithms and data structures
- programming languages
- architecture
- numerical and symbolic computation
- operating systems
- software methodology and engineering
- databases and information retrieval
- artificial intelligence and robotics
- human-computer communication

This is not the only possible division but it describes the domain very well and is used for different purposes, such as for classification of literature in Computing Reviews or for designing computer science curricula. Each of these subdomains has its elements of theory, abstraction and design that define the main features of it. The weights of theory, abstraction and design differ within each subdomain, that is, some subdomains are considered more practical than theoretical. As an example, algorithms and data structures, numerical and symbolic computation, and software methodology and engineering have a theoretical nature, whereas programming languages, architecture, and operating systems are aimed more at practice. The next discussion of some of these subtopics is based on a report of Denning et al. (1989).

Algorithms and data structures have a very strong theoretical background, which consists of computability theory, computational complexity theory, time and space bounds, levels of intractability, parallel computation, probabilistic algorithms, cryptography, and some supporting areas (e.g., graph theory, combinatorics, calculus, and semantics). Computer science students should be familiar with this theoretical background before they can fully understand how and why algorithms are designed the way they are, and how algorithms work. In addition, the theory of algorithms and data structures gives students fundamental concepts and guidelines to abstract and design their own algorithms. Algorithms and data structures form the basis for all computing, thus, students necessarily need to know them.

The theoretical basis of programming languages consists of formal languages and automata, Turing machines, Post systems and λ-calculus, formal semantics, and supporting areas, such as, predicate logic and temporal logic. This theoretical part often has a minor role in a computer science curriculum that prefers to concentrate on introducing different programming languages. This kind of attitude is harmful since without proper theoretical knowledge students are unable to understand the fundamental questions of programming language area, namely, what possible organisations presented by the language are, how these abstractions are implemented on computers, and what notation can be used effectively

and efficiently to specify what the computer should do. Programming languages are useful for giving substance to algorithms if the theoretical background is not forgotten.

The area of architecture aims at organising hardware into efficient, reliable systems, such as hardware units for fast computation (e.g., arithmetic function units), efficient methods of both storing and recording information and detecting and correcting errors, computer aided design (CAD) systems, and supercomputers (e.g., Cray and Cyber machines). Theory behind these aims is based on Boolean algebra, switching theory, coding theory, finite state machine theory, and supporting areas, such as queuing, reliability theory, discrete mathematics, and number theory. Architecture is easily misunderstood merely as pure hardware. This misconception should be corrected by emphasising the fundamentals of computer architecture in instruction.

Students often complain about the amount of mathematics included in computer science curriculum. It should be explained why they need such skills and knowledge. Numerical and symbolic computation focus on solving several problems, such as how we can accurately approximate continuous or infinite processes by finite discrete processes, how rapidly a given class of equations can be solved for a given level of accuracy, and how symbolic manipulations on equations (e.g., integration, differentiation, and reduction to minimal terms) can be carried out. The theories needed for understanding the meaning of and reaching these aims are number theory, linear algebra, numerical analysis, nonlinear dynamics, and as supporting areas calculus, real analysis, complex analysis, and algebra. Students need to learn at least the basics of these theories in order to be able to understand, design and implement algorithms solving mathematical problems. To quote Repo (1996), "a deep conceptual understanding is important in the process of learning algorithmic skills."

Database and information retrieval systems emphasise several topics, such as choosing appropriate modeling concepts to represent data elements and their relationships, combining basic operations into effective transactions, translating high-level queries into high-performance programs, protecting data against unauthorised access, disclosure, or destruction, and indexing and classifying text for efficient retrieval. The theoretical basis of this subdomain is also very extensive including relational algebra and relational calculus, dependency theory, concurrence theory, statistical inference, sorting and searching, performance analysis, and, as supporting theory, cryptography. Database and information retrieval systems are needed more and more in everyday life. Therefore, computer science students have to know this area well if they want to design and implement even better systems in future.

Computer science is a versatile domain that cannot be taught using solely one method. Some subdomains are clearly theoretical, others practical or mathematical, though all these aspects can be found in every subdomain. Therefore, a mixture of these different aspects needs to be considered when designing computer science curricula.

4.3 Instructional criteria

Computer-aided instruction should be student-centered and based on student's needs. Consequently, CAI courses should be interactive and controlled together by a student and computer. These demands, when they are fulfilled, promote student's motivation, whose importance as a part of successful instruction is strongly emphasised in many human learning theories. Another thing to remember is that instruction should be based on students' previous knowledge. This is important because then students are able to connect new information to their existing knowledge which strengthens their learning.

At first it should be confirmed the general view and basic concepts of the instructed domain are familiar to students. Together these two facilitate students' learning by offering them an appropriate knowledge base on which they are able to combine novel information. When students have a good general view of the domain they have better possibilities to recognise the connections between the domain and the novel information. As a result, every CAI course should clearly announce the prerequisites needed to understand the contents of the course. As an example, an introductory CAI course on the domain could give the required general view on the domain and its concepts, and no prerequisites on this domain should be necessary for understanding this course. Other CAI courses on subtopics of that domain could announce that the introductory course, or knowledge equivalent to it, and possibly some other prerequisites are needed to understand the contents.

The desired learning outcomes of the course should be clearly stated. In addition, students should be told how to recognise that the expected learning has happened. For example, students are informed that they have to pass an exam based on the presented CAI course material.

Another important thing would be to give a good general view of the content structure of the CAI course itself. This kind of content-dependent guidance assists teachers to choose appropriate CAI material. Students should be provided with an explanation of the connections to other subtopics of the domain and a summary of the contents of the course. Explaining the connections facilitates students to locate the novel information on a right place in their minds. The summary of the contents prepares students to receive the forth-coming information. Students should also be provided with technical guidance to use the course, for example, in a form of a guided tour.

Contents of a CAI course should be highly relevant to the instructional aims the course is having. Since it is wise to simplify the display content, as Chabay and Sherwood (1992) have stated, it is necessary to present only the most important things. However, oversimplifying must be avoided in order that the contents are not divided into irrelevant pieces.

Interaction between students and the course contents increases students' motivation because they get the feeling that they control the course flow. One possibility to increase

interaction in a CAI course, as Borsook and Higginbotham-Wheat (1991) point out, is to ask students questions, whenever it is possible, instead of telling them facts. While using questions one must avoid dead ends, where students are stuck if they do not know the correct answer. Students must be provided with either the correct solution after a few tries, or hints to find the correct solution, or possibility to back out the question. Interaction is discussed also in the user interface criteria section.

Feedback, which plays an important role in a CAI course, can be divided into two categories: instructional feedback and technical feedback. In principle, feedback should be given as a response to each action or performance of students, and it should be given immediately. Instructional feedback contains all feedback related to the subject domain and the contents of teaching material. Technical feedback, which is discussed in more detail in the user interface criteria section, is a response to practical matters, such as problems with the computer environment. The type of instructional feedback depends on the situation it is given. Different situations are, for example, students answering questions or students asking for instructional help. Appropriate feedback to students' answers are to tell the correctness of answers, give hints or instructions if the answers are incorrect, or give the right answer if students have failed too many times. Feedback to the instructional help request should provide assistance for correct performance. Implementation of instructional feedback is technically an extremely difficult task, since there exist several alternative possibilities how to response. Implementing of all possible responses is impossible, and choosing the most appropriate responses is difficult, as well.

Repetition is necessary for learning. Therefore, a summary at the end of lesson assists students to remember the essentials, or references to previously learned concepts or methods clarifies relations between these objects. Repetition should be used for strengthening students' learning. Recall is useful when connecting new concepts to students' existing knowledge by creating associations between them.

Students should have a possibility to practice their new knowledge and skills. Independent questions that have responses, may not yield the expected result, since students can easily guess the answers. If students are expected to give answers longer than just a few words, checking the answers becomes difficult, since the tools used for interpreting free-form answers are relatively restricted and not very efficient. More complicated means of practicing are problem solving tasks which force students to combine several pieces of knowledge and skills in order to be successful. Other expedient means are simulations and demonstrations in which students are able to try different combinations of parameters and observe consequences. When using these latter means, students should be given a well-specified target or a task to be fulfilled in order to avoid experimenting fortuitously.

Evaluation of students' learning outcomes gives information not only to students themselves and teachers but also to the authors of CAI courses. Evaluation information

should indicate to students the new level of their knowledge and skills so that they are able to compare with their previous capabilities. Teachers should use evaluation information to find possible defects in students' learning outcomes, and provide students with supplementary material if it is necessary. The authors of CAI courses should be informed if students systematically fail to obtain expected learning outcomes, since that indicates that some very fundamental things are wrong within the CAI course. A typical form of evaluation is an examination, which can be conducted either using traditional means or a computer.

4.4 Criteria for user interface

User interface is a very important part in a CAI course, since it is the only means for students to communicate with the computer. On one hand, the user interface should provide all necessary information for students at any time, but on the other hand it should be kept simple in order that students do not lose the essential information.

These interface criteria are influenced by ideas of Chabay and Sherwood (1992) and Ravden and Johnson (1989), who give guidelines for creation of educational applications and evaluation of human-computer interfaces, respectively. The advice presented by Chabay and Sherwood (1992) is based on several years of their experience designing CAI materials for various domains, and include guidelines for user interface design. Ravden and Johnson's (1989) practical method for evaluating usability of human-computer interfaces consists of nine criteria, which are visual clarity, consistency, compatibility, informative feedback, explicitness, appropriate functionality, flexibility and control, error prevention and correction, and user guidance and support. Some of the ideas of Chabay and Sherwood, and Ravden and Johnson are used with slight modifications in our criteria.

Interactivity plays an important role in an efficient user interface. It is discussed in more detail in the following section. Text, color, graphics, and animation are elements of which user interfaces are built. These elements are discussed later in this section.

4.4.1 Interactivity

Selnow (1988) has characterised interactivity as an interpersonal communication in which messages must be receiver-specific, message exchanges must be response-contingent, and the channel must provide for a two-way flow of information to accommodate feedback. Human-to-human models introduced by Berlo (1960) and Shannon and Weaver (1949) can be regarded as the theory underlying interactivity. More important is how interactivity can be used in CAI courses. Borsook and Higginbotham-Wheat (1991) have listed the following variables that they consider as the key ingredients of interactivity:

- immediacy of response (e.g., face-to-face conversation vs. mail),
- non-sequential access of information (e.g., presentation paying attention to needs of the audience vs. presentation without any breaks for questions etc.),

- adaptability (e.g., talking to a doctor or talking to a child),
- feedback (e.g., driving a car with open eyes instead of blindfolded),
- options (e.g., natural conversation between two people vs. conversation permitting the use of only few words),
- bi-directional communication (e.g., a normal telephone connection vs. a telephone connection for one-way conversation), and
- grain-size (e.g., possibility to interrupt an action vs. no responsive action before a certain state).

These variables should be applied in a reasonable mixture in a CAI course. The amount of each variable depends on the subject domain and the type of CAI course. As an example, adaptive systems and simulations use these ingredients of interactivity much more than CAI courses of other types.

The amount of interactivity reflects how the control over a CAI course flow is divided. Borsook and Higginbotham-Wheat (1991) claim that the optimal interactivity occurs when there is a balance of control between the student and the computer. A course is highly interactive if the student and the computer control the flow of it together. If either of them controls remarkably more than the other the amount of interactivity decreases. Therefore, a CAI course controlled solely by either a student or a computer has the lowest possible interactivity.

There exist several practical ways to enhance interaction, such as ask instead of tell, prompt direct manipulation, avoid typing, avoid mixing input modes, wait for students, provide instructions, and explain the interface (Chabay and Sherwood, 1992). Asking instead of telling forces students to give up their passive role and participate actively in the learning process. Direct manipulation enhances students' own thinking about the task. Typing should be avoided if there are available other more appropriate methods, such as pointing. Most students are not skilled with typing, which is harder than pointing. A mouse is the most usual pointing device, but not the only one. A keyboard can also be used as a pointing device, for example, by using arrow keys or the spacebar. Mixing different input actions for the same input should be avoided since moving by compulsion rapidly between input devices can be annoying. Waiting for students means that they should be given time to proceed on their own pace. Timed pauses should be used only when they are necessary, such as in animation. Instructions should be terse and fill a portion of a display for a task. In addition, it is better to show how to do than tell what to do. Thus students need not translate written instructions in one context into actions in a different context. Teaching the interface actions improves students ability to communicate with the computer. If these actions are taught properly at the beginning of the course, students attention can be concentrated on the teaching domain solely.

All feedback given to students should be highly informative since the display size is limited and, consequently, displaying unnecessary information should be avoided. Students should be aware of the following things:

- where students are in the application,
- what actions they have taken,
- whether these actions have been successful, and
- what actions should be taken next.

An evaluator should examine how these matters have been considered. In addition, availability of all necessary options and the quality of system feedback should be investigated.

Flexibility is an important feature of an application. Students should be able to undo and redo actions, look through a sequence of displays in either direction, move to different parts of application as needed, and name and organise information that may need to be recalled at a later stage. Availability of all these actions should be checked. Further, study of implementation of error prevention and correction should consist of checking how to correct errors in inputs before they are processed as well as after they have been processed, and checking the quality of the diagnostic information that application offers in an error situation.

User guidance and support should be informative, easy to use and relevant. It should be available both on the computer and as a paper document. Evaluated things include examining availability of help, clarity of help information, and relevancy of help information.

4.4.2 Display elements

Attention should be paid to several elements when designing a user interface. These elements include not only text, pictures and animation but also the connections between these display elements, usage of colors and highlighting, and how displays appear to students. Good general rules in display design are to keep things simple and leave blank space around elements.

Text

Authors of CAI courses should never use text lay out similar to books, since it is not an effective way to present things in a computer format. Instead of that, computers, as dynamic tools, offer new possibilities for presenting and making use of text, as Chabay and Sherwood (1992) put it. They have listed the following suggestions how to utilise text in a most effective way in a CAI course:

- Text should be used wisely for presenting basic information, clarifying and explaining pictures and diagrams, and providing both general and specific instructions.
- Instead of usual paragraphs, there should be used words and phrases with plenty of white space.

- Text and pictures should be mixed by inserting a few well-placed words, such as labels for the axes of graphs and important components of diagrams.
- One should strive for clarity to minimise chances of misunderstanding.

Visual elements

There exist several reasons why communication between the user and computer can rely on visual elements, such as pictures, animation and diagrams, rather than textual information. Bergin et al. (1996) have presented a collection of ten motivating factors for instructional use of visualisation tools as follows:

- Complex concepts can be clarified through the use of pictures: using only verbal expressions to teach complex concepts often requires a formalism that intimidates students.
- Visualisation is an alternative presentation model: students learn certain concepts better by thinking visually about them.
- Visualisation is a hook which can be used for grabbing students' attention: it has similarities with popular forms of entertainment.
- Instructors are able to cover more material in less time by using visualisation tools: demonstrations and examples presented in lectures can be automated by using the combination of visualisation tools and large-screen projection devices.
- Students' understanding can be increased by using good visualisations: an interactive visualisation gives feedback on correctness, lets students to proceed at their own pace, shows changes pictorially in an algorithm, and shows the outcomes when different data sets are input to an algorithm.
- Visualisation encourages modes of learning that instructors want to see in students: the use of highly interactive simulation software packages that support visualisation, stimulates and clarifies most of the concepts for students.
- Visualisation can be used for handling students with different backgrounds: including optional problems in assignments gives challenges to the more advanced students.
- Visual debugging: simplifies debugging process and removes some of the frustrations, thus rendering the whole process more tractable.
- Visualisation allows teachers to capture effective manual classroom techniques: automated illustration of procedures involving two or higher dimensional structures saves time, reduces errors and increases the clarity of classroom presentations.
- Visualisation can be broadened to perceptualisation: it is by no means limited to 'visual images.'

Visual presentations are worth using almost in every instructional situation, for example, in teaching concepts and algorithms of computer science. According to Bergin et al. (1996), the type of visual presentation used depends on the topic and cost effectiveness. Pictures are useful for both small amounts of data and very simple data structures. In addition, pictures should be used when the relationship of objects is important, but movement is not needed.

Animation should be used for presenting large amounts of data, complex data structures, or showing how the relationships between objects change over time. The cost benefit of developing a picture determines the type of created picture or animation, that is, is the picture

hand drawn or computer generated. Bergin et al. (1996) state that non-technical animations can be effective, may involve less time in their creation, and may involve the class with more interaction.

Animation can be used in instruction not only for presenting but also for doing things, for example, students learn by writing programs with animation. For creating animations there exist several animation tools, such as Aladdin (Helttula et al., 1989 and 1990), BALSA (Brown, 1988), DYNALAB (Boroni et al., 1996), Eliot (Lamminjoki et al., 1995; Lahtinen et al., 1996), GAIGS, Tango (Stasko, 1990), XTANGO (Stasko, 1992), and Zeus (Brown, 1992).

Connections between display elements

Organisation of different types of information affects the visual clarity of an application, since text and visual elements support each other when their connections are carefully designed. According to Chabay and Sherwood (1992), authors should connect both text with graphics and graphics with text. The first means that authors should avoid describing things that are better shown via graphics, including tying together pieces of text. Pointing graphically to the relevant words in a cited passage is better than attempting to describe their location verbally. The latter, connecting graphics with text means that authors make connections among the visual elements using text and sequencing (e.g., labeling axes of a graph). This facilitates understanding the relations between display elements. However, authors should remember to keep schemes and pictures clear.

Colors

Authors should be very careful when choosing and combining colors to CAI applications, since depending on authors' abilities to apply them, colors can help or hinder a design. Colors based on authors' personal preferences should not be used unless authors understand how to apply them to problem solving, since, to quote Shubin et al. (1996), "color can appear to advance or recede, make boundaries vibrate or blend, emphasise or blur shapes, and cause other colors to change." Visual clarity of an application suffers from careless use of colors.

Highlighting

Highlighting, like colors, affects the visual clarity of an application. Since highlighting is a useful tool in interface design, authors should remember several things in order to be successful in a design process. As Chabay and Sherwood (1992) point out, highlighting should be used for providing a focus on the most important things. Possible ways to achieve this result include:

- using italics,

- blinking text, arrow, etc.,
- using borders around a major item of text,
- pointing at something,
- using "inverse video," and
- using distinctive colors.

In addition, highlighting should be used sparingly, since if everything is highlighted, nothing is important. The use of too many colors or too many text styles in one display can be very disturbing and diminish readability of information.

Important display elements should be made big enough and be placed at the center of the display. Less important things should be left to the periphery or behind menus from where they can be invoked if needed. Irrelevant elements, that are no longer needed, should be removed.

Display-building

There exist several ways how a display can appear to students. The basic rule is that new information should be near the most recently displayed information. This adjacency facilitates students to find new information on the display and makes a complex display understandable. Authors should remember the following considerations when building up a display a little at time (Chabay and Sherwood, 1992):

- Displays should be built up piece by piece to give students time to identify and comprehend the components of a display.
- At the same time authors should both maintain visual context and avoid disorder. Finding a balance between these needs is a necessary, though difficult task.
- Authors should use both single and multiple "pages," since they serve different purposes. Single page applications have a strong contextual framework visible at all times, whereas multiple pages provide a more flexible framework, for example, an optional extended discussion.
- Long texts should be broken down, since building, for example, a list piece by piece maintains emphasis on the most recent item. At the same time it also maintains the context, since the previous items are still visible.

It is also good if students are able to choose the rate at which the display is built up.

4.5 Pragmatic criteria

There exist several points of view to be considered when CAI courses are intended to be used as a part of instruction. These considerations include hardware, software and human resources, and both teachers' and students' attitudes and readiness to use computers.

Availability and compatibility of hardware are important things to consider. Computer resources should be adequate for proper use, that is, there should permanently exist enough computers in proportion to the number of students. Any special arrangements (e.g., transporting one or more computers from one classroom to another each time they are needed

in instruction) should be avoided, since such activities take too much time and effort in the long run and, consequently, they may reduce teachers' willingness to use computers. Compatibility of hardware should be checked when running a CAI application requires use of external equipment, since there do not exist standards for all peripherals. Such external equipment must be compatible with the intended instruction computer.

Like hardware requirements, software requirements of a CAI application also need to be considered. As an example, a CAI application may run only in a computer with a certain operating system, or it may need a specific external application (e.g., text processor or spreadsheet calculator) to function properly. In addition, properties of an application may be adjusted to properties of a certain processor, mainly its speed of execution. In such cases, the application may function in a different way than it was intended, for example, animation may be executed so fast that it is impossible to follow it. Therefore, the computing environment requirements of a CAI course should be announced clearly in course documentation, which should inform users of all technical information related to installing and running the CAI course.

Human resources concern both teachers and other staff involved with CAI. Teachers must be available since students need guidance not only with the subject domain but also for becoming familiar with the CAI application and the computer. Other staff consists of persons who are responsible for the maintenance of computers and applications. Such personnel is needed since all teachers may not be able to install applications or solve problems occurring in computers.

Teachers' and students' attitudes and readiness to use CAI affect instructional computer use. Attitudes to computers should be positive both among teachers and students, otherwise the use of computers cannot be very efficient and effective. Readiness to use CAI means that users have appropriate skills and knowledge to interact with computers. Lacking such skills complicates the learning process, since users must concentrate on tools, rather than contents. Therefore, teachers and students should be motivated by telling them why and how computers can be useful in instruction, and they should be taught to use computers effectively. Teachers should know well CAI applications used by students in order to be able to assist students when they have problems. Furthermore, students should have a guided tour to get familiar with the CAI course. This tour should cover the explanations of all features and tools in the course. Students should also be able to try different features during the tour to get used to them.

5 Test case: COSTOC

Test case: COSTOC

Some CAI courses on computer science were chosen to be analysed to obtain detailed information about how computer science has been taught using computers. We also wanted to test our criteria described in chapter 4. The results of the analysis may be useful when designing new CAI courses on computer science and when designing properties of authoring tools supporting special features of computer science.

This chapter describes how our evaluation criteria are applied into practice. An analysis method based on the criteria is presented. Results of the analysis consist of the general results and the results of each tested course which are presented in corresponding order. Results of each COSTOC course are discussed separately because the main interest was to study how each topic has been taught using computers rather than to compare different courses to each other. The results of each tested course are presented in two groups: course contents and instructional support.

5.1 Analysis method

The target of this analysis is to find out how computer science has been taught using computers, measured by the criteria set in the fourth chapter. In addition, we want to evaluate the quality of the selected CAI courses on computer science. Analysis consists of investigation of internal structure (i.e., instructional support and topics related both to contents and organisation of courses) and external structure (i.e., user interface) of tested courses. Both internal and external structures need to be examined in order to get a balanced view on each tested course and to be able to answer the research questions of this study.

The analysis method, which is designed by the author of this work, is strongly based on counting different structures of tested courses. These structures can be frames and their

contents, like in tested courses, but they can be nodes as well, like in hypertext applications, or pages, like in multimedia applications. This analysis method can be applied in several CAI applications of different types since it is enough that there exist some structures and their contents that can be examined and counted. There are four major subjects of interest that are examined:

1. course contents
   - content types of information structures
   - illustration of information structures
2. instructional support
   - amounts of different structure types
   - topics emphasized in instructional criteria
3. user interface
4. pragmatic matters

5.1.1 Analysis of course contents

Analysis of course contents is based on criteria for the subject. This part of the analysis aims at finding out how each of the tested courses tries to teach the topics, and finding out the relevancy of the course contents to the aims. The main analysis method is to examine the contents and illustration of information frames.

Contents of information structures

A detailed analysis of the contents in information structures must be done in order to find answers to three questions; namely, what kinds of things they are teaching, how these things are taught, and why these things are taught that way. The contents of each information structure are examined and different content types are counted.

Each information structure may consist of several types of contents. Due the fact that computer science is a mathematical science, it is expected that information structures will consist of definitions, equations, formulas, and proofs. Furthermore, programming code and pseudo code are expected since programming is an essential part of computer science, and the tested courses should represent evenly the domain of computer science. In addition to these content types, some other types are likely to appear.

Illustration of information structures

The illustration of information structures is examined because illustration helps students to understand new concepts and phenomena. The common illustration types, namely, examples, pictures and animation, are expected to be found. The importance of examples is based on their ability to illustrate new concepts by connecting them to old ones by using, for example, comparisons. Some concepts and phenomena are easier to understand in a visual

form than in a textual form. Pictures and animation are excellent for presenting material in a visual form. Examples, pictures and animations are counted.

Relevancy of the course contents to the aims

After examining the tested course completely, relevancy of its contents to the instructional aims is evaluated. This is done by comparing the aims to the contents.

5.1.2 Analysis of instructional support

A CAI application is supposed to instruct something and, thus, there must support for human learning. A natural place for instructional support is in information structures. Analysis of instructional support consists of examining how courses fulfill the instructional criteria presented earlier in this chapter by counting the number of different structure types, by studying what kind of learning support is provided, and by comparing how structures are illustrated. The applied analysis methods are described in groups emphasised in the instructional criteria:

- motivation,
- general view of the course contents,
- prerequisites,
- informing of the learning aims,
- structure of the course,
- guidance to use the course,
- interaction,
- instructional feedback,
- repetition and recall,
- practicing new skills or knowledge, and
- evaluation of learning outcomes.

Motivation

Means of motivation are examined and the amount of motivation in information structures is measured. It is expected that motivation is found evenly throughout the lessons.

General view of the course contents

It is examined how the general view of the course contents is described. This description is expected to be found both in the documentation and the on-line help.

Prerequisites

The appearance and clarity of prerequisites are investigated. Prerequisites should be found both in paper documents and the on-line documents.

Informing of the learning aims

Appearance of information on the learning aims is studied. Such information is expected to be found in the beginning of each lesson.

Structure of the course

The structure of the tested course is measured by counting the number of different structure types. Each tested course is expected to contain at least three different structure types: navigation, information, and question structures. The numbers of structure types are counted in order to find out the proportions of each type. These proportions indicate the general organization of a course and the importance of different structures within the course. The presented percentages and ratios, which are based on opinions of the author (see Table 5.1), are normative, since the extent of contents of information structures vary from course to another and the differences can be very big. These percentages are our approximations for a balanced CAI course. As far as we know there does not exist percentages for this kind of purpose. Thus, we make one suggestion based mainly on our own experience. The percentages are discussed in detail below. The percentage of question structures does not tell the whole truth since one question structure may contain several questions, or there may exist several question structures for one question. Therefore, there are no percentages for question structures presented.

Navigation is an important feature in a CAI course of any domain because it gives students the control, that is, freedom to choose what to study. If the percentage of navigation structures is low, that is, under 10%, it is possible that the course is too strongly controlled by the computer, that is, students are forced to follow a certain route through the course material with only minor choices. On the other hand, if the percentage of navigation structures is high, that is, over 25%, there exists a risk that the course is fragmented and, thus, students have difficulties to follow the flow of the course.

The ideal percentage of information structures is between 60% and 70%. If it is less than 55%, the contents of the course is inadequate and students' learning remains superficial. Percentages over 75% dominate too much the course structure and students' interest and motivation may vanish.

Instead of the percentages of question structures, the amount of questions must be considered. If the ratio between the amount of information structures and the amount of questions is more than four, there are too few questions for testing students' knowledge and learning may remain inadequate. If the ratio is less than two, questions appear too frequently and students are not able to concentrate on the essence.

Table 5.1: Limits of structure types in percentages.

Structure type	Too low	Ideal	Too high
Navigation	<10%	15%-20%	>25%
Information	<55%	60%-70%	>75%

Guidance to use the course

It is checked whether there exists both content-dependent and technical guidance to use the course and how it is implemented. Guidance is expected to be available in the beginning of the course though it should be accessible as needed throughout the course.

Interaction

This is mainly described in the subsection dealing with the user interface analysis. Only the instructional interaction is examined in this subsection.

Instructional feedback

The types of instructional feedback are examined and the quantity of each type is counted. Instructional feedback, like motivation, is expected to be found evenly distributed throughout the lessons.

Repetition and recall

The types of repetition and recall are investigated and the quantity of repetition and recall is counted. Repetition is expected to be found at least in the end of each lesson. Recall is expected to be found in situations where students are provided with new information that is based on previously presented concepts or phenomena.

Practicing new skills or knowledge

The type and amount of practice are investigated. Practice should be found in every tested course but not necessarily in every lesson.

Evaluation of learning outcomes

The appearance of the evaluation of learning outcomes is examined. It should be found at the end of tested courses.

5.1.3 Analysis of user interface

The appearance of the user interface is studied to find out its effects on learning. This part of the analysis is based on criteria for user interfaces. It is strengthened by the evaluation method designed by Ravden and Johnson (1989), which is chosen because it states some

important notions on user interface evaluation. Tukiainen and Lempinen (1994) have adapted Ravden and Johnson's method to be suitable for evaluating the usability of software. This latter method is also applied to the evaluation. Only the suitable parts of both methods are used.

Analysis of interactivity

Implementation of interactivity is examined by checking types of interaction, feedback, flexibility, error prevention and correction, and user guidance and support. The following things are examined:

- how interaction is enhanced (e.g., asking vs. telling, prompting for direct manipulation, avoiding mixing input modes),
- what kind of technical feedback does the application provide (e.g., what actions students have taken),
- type and nature of messages and instructions,
- clarity of requirements to take a particular action,
- clarity of the type of information needed to be entered,
- whether it is possible to undo or redo actions,
- whether it is possible to look through a sequence of displays in either direction,
- whether it is possible to move to different parts of application as desired,
- whether the user is able to choose the rate at which information is presented,
- whether the user is able to name and organize information which may need to be recalled at a later stage,
- whether it is easy to correct errors in inputs before they are processed,
- whether it is easy to correct errors after the inputs are processed,
- whether the system offers all necessary diagnostic information in an error situation,
- whether help facility is easily available from any point in the application,
- whether the help information is context-sensitive and presented clearly,
- whether the paper document provides an in-depth, comprehensive description, covering all aspects of the application, and
- whether the needed information is easy to find from the paper document.

Analysis of display elements

Analysis of display elements consists of examining implementationof text, visual elements, connections between display elements, use of colors, highlighting, and display-building. The following items are studied:

- how text is used (subject material, feedback, error messages, help),
- how visual elements are used (pictures, animation),
- what kind of connections there exist between display elements,
- how colors are used (text, background, highlighting, etc.),
- how highlighting is used (italics, blinking, borders, pointing, colors, etc.), and

- how displays are built-up.

5.1.4 Analysis of pragmatic matters

Pragmatic matters include checking the required hardware, software and human resources. Teachers' and students' attitudes and readiness to use computers affect how successfully computers can be used in instruction. Even though these attitudes and readiness are not directly involved with any certain CAI application, they should also be examined. The following pragmatic matters should be checked:

- what kind of hardware is available,
- whether the hardware is compatible with the intended CAI application,
- whether there exist some special software requirements with the intended CAI application (e.g., certain operating system version, external applications, etc.),
- whether there exist appropriate documentation for installation,
- what human resources are available (teaching personnel, installation personnel),
- what kind of attitudes teachers and students have towards computers, and
- whether teachers' and students' readiness to use the application is considered somehow (e.g., guided tour to use the application).

5.2 Background information of the analysis

In searching for test material, there were two problems. Firstly, it was difficult to find any CAI applications on computer science that had been used systematically in instruction. Secondly, most of the courses discussed in literature were laboratory experimentations and not commercial products and, thus, they have not been used widely. Real usage would have been important from the analysis point of view. COSTOC-courses were exceptions since they had been available commercially and several universities have used them, for example, University of Karlsruhe, University of Kuopio (Kopponen, Kasurinen and Linna, 1991), and University of Texas at Dallas. Therefore, they were chosen as the test material.

The COSTOC (COmputer Supported Teaching Of Computer science) project forms a special collection of teaching material because it has been one of the major attempts to introduce computer aided instruction on a large scale to support teaching at university level, mostly in the area of computer science. The COSTOC project started in 1985 in Technical University of Graz, Austria, and lasted 6 years until 1990 under the leadership of professor Maurer.

The idea of COSTOC project is unique because it gathered the knowledge of internationally prominent experts in uniform courses on diverse topics in computer science. There exist more than 20 courses for teaching computer science or related subjects. The topics include, for example, operating system concepts, computation and automata, trees, sorting techniques, systematic programming, computer networks, syntax-analysis, digital

logic, database systems, cryptography and data security, and several courses on programming languages (Ada, C, Lisp, Pascal and Prolog).

COSTOC courses were selected for this analysis because they are the result of an extremely consistent project of designing CAI courses on computer science, they represent various topics of computer science and they are considered to have high-quality contents. The analysis concentrated on both the external and internal features, i.e., how the user interface has been implemented (e.g., the lay-out of contents on the screen), what kind of teaching strategies have been used (dialog, simulation etc.), how learning has been supported (motivation, repetition, questions etc.). Furthermore, the analysis tried to find out the special features of computer science (e.g., the mathematical nature with clauses and proofs, programming code) and how they are taken into consideration within COSTOC courses.

5.3 On COSTOC courses selected to analysis

Five COSTOC courses, namely, Computation and Automata, Cryptography and Data Security, Introduction to Database Systems and the Relational Data Model, Sorting Techniques, and Systematic Programming, were selected for detailed analysis. These courses were chosen because they represent fundamental topics in computer science. In addition, we had difficulties to find such CAI courses on computer science that both instruct and test students' learning. The chosen courses that were selected from over twenty COSTOC courses represent different domains of computer science. Each of the chosen courses will be shortly described after the general description of COSTOC courses.

5.3.1 General information about COSTOC courses

COSTOC courses consist of lessons. A full course is composed of 11 lessons: one for introduction and ten for instruction. A half course has six lessons. The introductory lesson (lesson 0) usually consists of the following topics:

- aims of course,
- prerequisites,
- level of course,
- contents of course,
- structure of course,
- how to use the lessons,
- references, and
- samples of lesson fragments.

Lessons are divided into chunks of information which are called frames and identified by frame numbers. Each frame may contain text, graphics, and animation. Within a frame there may be pauses where nothing happens for a certain amount of time or until student interrupts by clicking the mouse or by pressing any key. There are three types of frames: navigation

frames, information frames, and question frames. Navigation frames are only for choosing where to go or what to do. Title frames and end frames are considered as navigation frames though there are no choices in them. Information frames are used for representing teaching materials. The contents of information frames are discussed more later in this chapter. Question frames contain questions for students. All choices within the frames can be done through the keyboard or the mouse. A flow of a typical COSTOC lesson is described in Figure 5.1. Students start from frame 1 and are able to select any of the sections within the lesson. Arrows between frames depict students possibilities to move forward and backward. In addition, the navigation option via a button is available all the time.

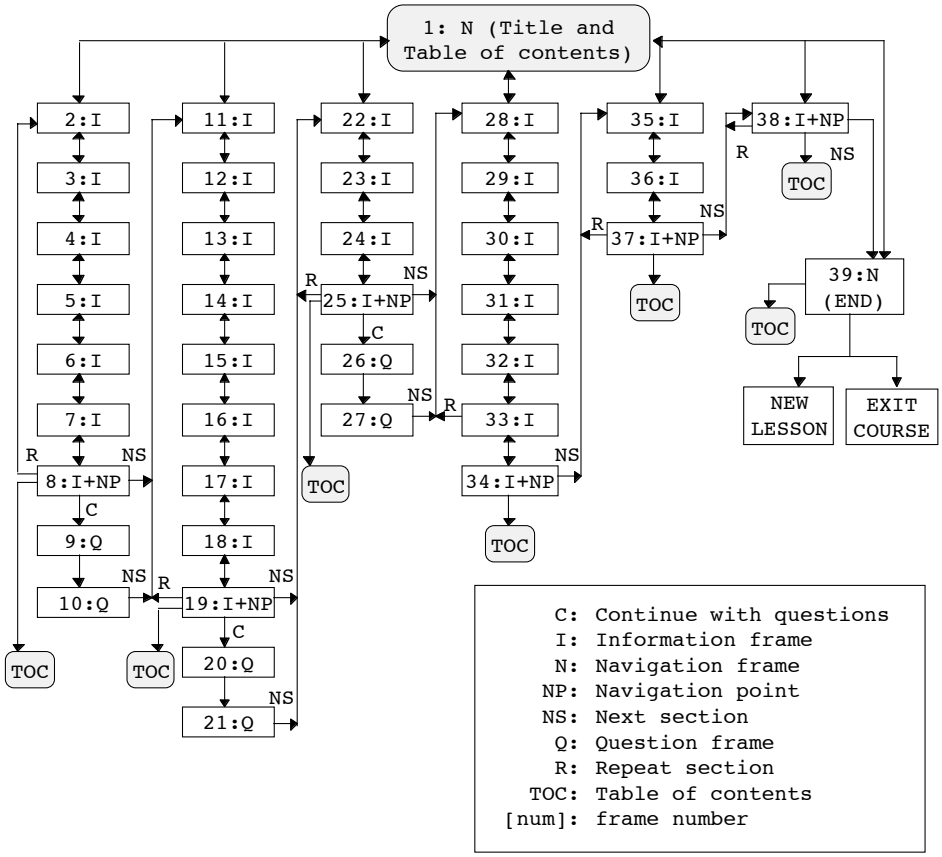

Figure 5.1: A typical lesson flow of a COSTOC course.

All COSTOC courses have a similar user interface, which consists of two button bars, a main window, an annotation window and the status information (see Figure 5.2). Button

bars are designed for navigating, setting options, and controlling the course in general. The main window is for representing the course material. The annotation window is for displaying teachers' comments on material or other information for students and for recording and displaying students' private notes. Status information tells the status of various options.

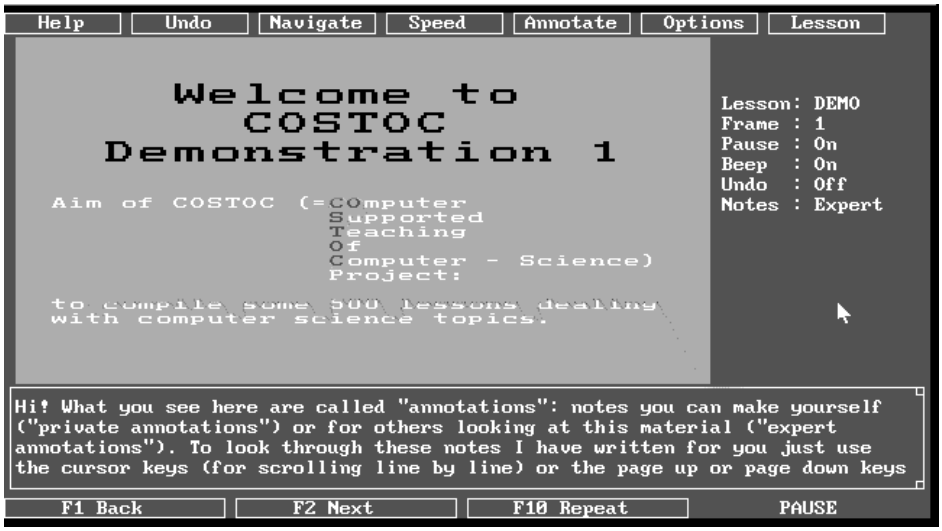

Figure 5.2: The basic display of COSTOC courses.

The upper button bar, which is placed on the top of the screen, consists of seven buttons labeled `Help`, `Undo`, `Navigate`, `Speed`, `Annotate`, `Options` and `Lesson`. Each button may be invoked by clicking it or pressing the `ALT` key together with the capital letter of the label. The functionality of each button is listed next:

`Help` button gives information of navigation and control features. It is context dependent, i.e., it gives information depending on the situation in which it is activated.

`Undo` button tells the status of undo function and advice how to activate it.

`Navigate` button offers three possibilities: `goto frame`, which asks where to go; `mark frame`, which sets a bookmark on current frame and shows it in the status information list; and `jump to bookmark`, which changes the marked frame to current frame.

`Speed` button allows adjustment of animation speed.

`Annotate` button opens annotation window for writing students' notes if private notes are on. If private notes are off button gives instructions how to change them on.

> `Options` button has four options to toggle: `pause`, `beep`, `undo`, and `switch notes`. `Pause` changes the proceeding control from student to computer. `Beep` controls sounds which be used to alert students to particularly important points in a lesson. `Undo` allows students to go back step by step, as much as they want, through what they have seen before. `Switch notes` option changes from expert notes to private notes and vice versa. The path of the note file is asked in every change.

The button bar on the bottom of the screen is sensitive to the current frame. It changes depending on the frame that is active. The possibilities are listed next:

> When the current frame is a navigation frame, there are `F1 Back` button and a set of numbered buttons for branching. `F1 Back` leads back to a frame that the author of the lesson has considered the "logical predecessor," which may not be the last frame the student has seen. The number of buttons depends on the amount of possible choices within the navigation frame.
>
> When current frame is an information frame there are three buttons, `F1 Back`, `F2 Next`, and `F10 Repeat`. `F1 Back` button functions as described above, `F2 Next` button leads directly to the next frame (ignoring potential material still going to be presented on the current frame), and `F10 Repeat` button repeats the current frame from the beginning (including the annotations).
>
> When current frame is a question frame, there are the same three buttons as in information frames. `F2 Next` and `F10 Repeat` function as described above, but `F1 Back`, like `F2 Next`, skips the current question and enters the next frame (question or information frame). In addition, on the right side of the screen there are `ESCape` and `Enter` buttons for skipping and answering the questions, respectively.

The main window, which is below the upper button bar, is for displaying the lesson material. This is the area that the authors use for designing instruction. The size of the area is 24 rows and 40 characters on each row. Authors may compose their lessons to contain text, graphics and animation as they wish. There are different text sizes and special characters available.

The annotation window, which is placed below the main window, is for displaying notes. Notes are frame-sensitive, that is, they change when the frame changes. It is possible to have several expert and private notes for each frame since whenever a lecturer or a student starts to make notes, they are asked to specify the file where the notes will be saved. In the mode Private notes, if Expert notes are available, this is indicated on the right side of the screen. The student can switch to Expert notes to read them, if desired.

The status information is placed on the right side of the screen. It identifies the current lesson and the active frame, the status of `pause`, `beep` and `undo`, and the type of notes in use (expert or private). It also shows if any frame is marked as a bookmark.

5.4 General results of the analysis of COSTOC

The general results of the analysis of COSTOC courses are divided into four groups, namely, course contents, instructional support, user interface, and pragmatic criteria. The results presented in this section describe properties that are common to all tested courses.

5.4.1 Results of the course contents analysis

First, the content types of information frames are presented followed by the description of the illustration of information frames and the relevancy of the course contents to the learning aims.

Content types of information frames

The three general content types of information frames in COSTOC courses are analysis, definitions, and equations and formulas. In each course, analysis is considered as analysing different properties or characteristics of the subject, for example, in Sorting Techniques course explanations concerning space or time complexity of a sorting algorithm is considered as analysis. Most often analysis is verbal, though in some cases pictures and animation are utilised to illustrate it. Definitions explain new concepts and phenomena using both verbal and visual expressions. Equations and formulas are classified into the same type since they both consist of mathematical expressions.

In addition to the three general content types, other types exist. Computation and Automata course contains productions, programming code and proofs. Introduction to Database Systems and the Relational Data Model course contains programming code and relations. Systematic Programming contains programming code. Sorting Techniques course contains programming code and proofs. The proportions of content types are presented in percents in figures.

Illustration of information frames

The information frames are illustrated using examples, pictures, and animation. Examples are textual, numerical or visual (i.e., pictures or animation). The real amount of examples is larger than the stated amount in the numerical results of each tested course since many information frames contain more than one example. Pictures are motionless whereas animation consists of pictures that have moving elements.

Relevancy of the course contents to the aims

Course contents have a high relevancy to learning aims in all those tested courses in which the learning aims were stated.

5.4.2 Results of the instructional support analysis

The results of instructional support analysis are presented in the same order as described in the analysis method, that is, structure of the course, motivation, informing of the learning aims, guidance to use the course, relevancy of the course contents to the aims, interaction, feedback, repetition and recall, practicing new skills or knowledge, and evaluation of learning outcomes.

Motivation

The learning support in information frames includes motivation, recall and repetition. In addition, there exist thinking points which are discussed in the section concerning with interaction.

In all tested courses, students are motivated by using explanations and both verbal and visual examples. Motivation varies a lot within the separate lessons of the courses. There exist lessons that have no motivation at all whereas some lessons have several motivating explanations or examples.

General view of the course contents

Course contents are described well both in paper documents and in the introductory lesson of each tested course.

Prerequisites

Prerequisites and the level of the course are stated clearly in paper documents and in the introductory lesson in three out of five tested courses. In one tested course prerequisites are stated only in the introductory lesson, and from one tested course prerequisites are totally missing.

Informing of the learning aims

Aims of the courses are not given in all tested courses. If learning aims are given, they are presented in the introductory lesson (lesson number 0) using dashed or similar lists.

Structure of the course

The structure type used is a frame. Each COSTOC course consists of three types of frames, namely, navigation, information and question frames. All frames that have menus, including the start and end frames, are considered as navigation frames. Furthermore, in some information frames there are navigation points, that is, there is a possibility to choose what to do next (e.g., skip the rest of an example).

The most common type of question is the multiple choice question. There are only few other types of questions, such as verbal or numerical questions. A student has one or two tries to answer a question depending on the course. Upon failure, the correct answer is shown.

Guidance to use the course

Content-dependent guidance is provided in four out of five tested courses, that is, introductory lessons contain instructions how to use the lessons. Technical guidance is provided only in the demonstration version of COSTOC courses in which it is implemented as a guided tour through the menus and options.

Interaction

The forms of instructional interaction in tested courses are questions and thinking points. The latter ones are places where students are actively asked to think the solution before proceeding. The idea of thinking points is good since it actively reminds students that thinking is very important and things are not always easy to understand.

Other results concerning interaction are presented within the results of user interface analysis (see Section 5.4.3).

Instructional feedback

Instructional feedback is found only in question frames. Feedback to students' answers is most often short verbal feedback. For multiple choice questions, feedback is provided if the given answer belongs to the group of displayed possible answers, that is, the answer is one of the choices. If the answer is not within the choices, no feedback or notice is given. For other type of answers feedback is given always.

Technical feedback is discussed in results of user interface analysis (see Section 5.4.3).

Repetition and recall

Repetition and recall are important methods for building the overview of a course. In many cases, repetition is a summary of the most important concepts. Recall, when it is used, is a reference to previously discussed topics or concepts.

Practicing new skills or knowledge

In each tested course, the only form of practice is via questions, which are most often multiple choice questions. The other question types are textual and numerical questions. Multiple choice questions are easy to implement, answer, and check the answer. The problem is that guessing is very easy, especially when there are only two choices. In such cases, we recommend not to use multiple choice questions. Verbal questions are also easy to create but answering and checking verbal answers are complicated. Students need a model to answer, for example, whether upper case and lower case are equal. Checking verbal answers is difficult since there might be typing errors, synonyms, missing words, extra words, etc. which all should be handled. This type of question is good since it forces students to think the answer more carefully than just choosing one choice. Numerical answers are also easy to check if only the result is required. If students have to explain how they got the result, checking becomes difficult. Nevertheless, that kind of question should be available for practicing since numerical answers with explanations demand students to use their problem solving skills.

Evaluation of learning outcomes

No attempt is made to consider evaluation of learning outcomes in any of the tested courses.

5.4.3 Results of the user interface analysis

These results consist of the results of interactivity analysis and the results of display elements analysis. Since the user interface is common to all tested courses, all results related to user interface analysis are presented here.

Results of the interactivity analysis

Enhancing interaction. Interaction is very simple in all tested courses. Students control moving forward within a frame by pressing any key excluding some special keys that are used for moving backward in a frame or moving between frames. No special methods have been used to enhance interaction. Cursor can be moved around the screen using either a mouse or a keyboard. Both methods are available always.

Information can be entered either using a keyboard or mouse. Both of them can be used when answering questions, performing actions and changing options. Since questions are most often multiple choice questions, the input information in those cases is simply one number. Numerical answers are obvious and textual answers are very short consisting of up to three words. A place for answer is showed by a blinking underline character. There is no special prompt for answers, though on the right side of the screen there appear two buttons, `ESCape` and `Enter`, when application is waiting for student's answer.

Technical feedback. There is some status information that tells students which lesson and which frame in that lesson they are studying. Previous actions, whether these actions have been successful or not, and what actions should be taken next cannot be seen anywhere. Feedback provided by the system is scarce since there exist merely a few error messages and limited help information. Feedback is only partly appropriate because in some places provided help is too general or even missing.

Messages and instructions. There exist only a few messages and instructions in COSTOC courses. The messages are most often error messages informing about a missing special character file. Instructions to perform a particular action are given in a short demonstration

program, which is the only place where they can be found. On the other hand, actions in COSTOC courses are very simple and quick to learn.

One peculiarity is the message that appears when a student has been using the tested course for quite a long time, telling that it is time to have a break. That is reasonable but the following action is shocking: the system shuts itself down without giving the student any chance to prevent it.

Clarity of needs to take a particular action. Because of the simple structure of user interface of COSTOC courses, all options are in sight of students during the lessons. This is good since students are able to control the system easily. Nevertheless, the clarity of the meaning of available options is not very good. For example, `help` option differs from the other options on the same button bar in its functions since it is applicable together with other options. This difference is not informed clearly.

The `Undo` function is useful but there exist some problems in its usage. When a student wants to change undo from `off` to `on` the system asks a path without explaining what is exactly wanted and why. In addition, if the confused student then wants to cancel this function it is not possible even though there is an `escape` button available. A similar problem appears when changing notes from expert to private and vice versa.

The `Navigation` function does not work properly. The problem is that when navigating from one frame to another using `goto frame` option, a student has to know the target frame number by heart since there is no list of frame numbers available. Another minor problem is that there can exist only one bookmark simultaneously.

The option for adjusting speed is unclear since there is no mention how speed is changed. Trying out the `speed` function reveals that it has something to do with the animation speed. Still it is a confusing function because it is not explained whether the speed of animation is changing, which would be the expected function, or the delay of the animation is changing, which really happens.

The structure of the system is clear since it is very simple. There exists only one screen which includes all possible buttons and options. Therefore, students should not have any problem knowing where they are.

Clarity of the type of information needed to be entered. Some questions to which textual answers are expected should have an example answer format, since without it there is a good chance for a misunderstanding. This is not a serious problem since most of the questions are multiple choice questions.

Possibility to undo and redo. Students are able to undo actions they have taken if they have earlier switched `undo` option on. `Undo` is implemented so that when `undo` is on, all screen images that a student goes through, are temporarily saved on hard disk until `undo` is switched off. This method slows down execution speed of a course slightly.

Possibility to look through a sequence of displays in either direction. It is possible to look through a sequence of screens either backward or forward by using `Back` and `Next` buttons. In addition, there exists a `Repeat` option which repeats the contents of the current frame.

Possibility to move to different parts of application as needed. Moving to different parts in a course means either moving within a lesson or between lessons. Moving within a lesson is possible by using `navigate` function which has two choices, namely, going to a specified frame or jumping to a bookmark. When a student wants to go to a specified frame, he has to give the number of it. Jumping to a bookmark works when there exist a marked frame where to jump. Moving between lessons is possible but a bit inconvenient. Students must be familiar with the way how directories and files are organized in the operating system to understand how to move to a new lesson. Some help is available on request.

Possibility to choose the rate at which information is presented. The execution rate of lessons can be controlled by the computer or by the student. It is always the student who decides which of the two has control. In a normal situation, a student controls the execution rate by using a keyboard or a mouse. In this case, frame contents are built piece by piece and student may take his time to read the contents. The computer has control when a student switches `Pause` option `off`. In this case, the execution rate is so fast that the student has no time to read the text or look properly pictures and animation. Therefore, the computer should

have control only when a student is looking for something special and quickly wants to browse already familiar material.

Possibility to name and organize information for later use. Recalling information is possible by using a bookmark. Drawbacks are that there can be only one frame marked at a time and the only reference to that marked frame is its frame number, which does not tell anything about its content. It is not possible to save a certain situation and continue later from that situation.

Correcting errors in inputs before and after processing. Errors may occur when students have to choose what to do or when they enter their answers to questions. In a case of choosing, invalid inputs are rejected and nothing happens. The situation is different when answering questions. If the input is wrong it can be changed before processing, or otherwise it will be judgedincorrect.

Diagnostic information in an error situation. The only real errors are occasionally appearing system errors which are caused by missing special character files. The error message informs that as a result of these missing files the screen appears incomplete. The name of the missing file is also reported. It is odd that sometimes a frame using some special character file appears complete and a few moments later, for example, when repeating the same frame, an error message is presented and the frame is incomplete.

Availability of the help facility. Help function is available almost everywhere in COSTOC courses. It is partly context-sensitive, that is, it gives specified instructions in different situations, for example, in adjusting the speed of animation, in changing `undo` option `on` or `off`, in switching notes between expert and private, or in moving to a new lesson. In some places it either presents one general instruction or no instruction at all. Help function is good and works properly.

Clarity and context-sensitivity of help information. Context-sensitive instructions, which are clear but not very detailed, can be found using the help button.

Quality and usefulness of paper documents. All novel users are supposed to go through the demo version of COSTOC courses since it is the only place where the user interface and its functions are describedproperly. Paper documents offer merely information concerning lesson contents, not the user interface. This is an obvious defect.

Results of the display elements analysis

Usage of text. A lot of text has been used to present the subject material in all tested courses. Occasionally there is too much text in one display, that is, from the top of the display to the bottom line (24 lines, 40 characters/line). Feedback text and text in error messages are usually very brief. In some cases, there should be more explanations. Help texts are longer than the other texts but not much.

Usage of visual elements. Schemes and pictures are simple and easy to understand though from time to time the system prints a message about a missing graphics file, which causes a loss of some special characters. Pictures and animation are used often and in most cases they are simple but effective.

Connections between display elements. Different types of information have their own places on the screen. There are buttons both on the top and the bottom of the screen. Status information is placed on the right side of the screen. Main part of the display is reserved for presenting teaching material. There is also an area reserved for expert or private notes. This organization remains unchanged in all COSTOC courses. Text and visual elements used in the main window are connected together with visual aids only seldom.

Usage of colors. Usage of colors is not consistent in all courses, namely, in Sorting Techniques and Systematic Programming courses both text and background colors in information frames vary a lot, which makes these courses look tangled. In some places the usage of colors is unsuccessful and makes reading difficult, for example, yellow letters on a red background.

Usage of highlighting. Information is highlighted using underlining, blinking and colors. Underlining disturbs reading the text slightly in some cases where the line spacing is narrow. Blinking is used for highlighting the cursor, which is an arrow, and for some animations. The blinking cursor is disturbing. Colors are used for highlighting titles, new concepts, pictures and

animation. The usage of colors for highlighting is most of the time good but there are occasionally too many colors on the screen at the same time and, therefore, it is difficult to know what is the most important issue.

Display-building. Information on the screen is mainly easy to see and read. Display is built-up starting from the upper left corner of the main window and proceeding to right and down. Display is built piece by piece under students' control. Sometimes display-building takes a long time if there is a complex picture to be presented. In such a case, students may try to hasten the build-up process but that is not possible.

In a few places there is need to display more information than can be placed in one display. In such cases the top part of the display is erased and the rest is lifted up to give space for new information.

5.4.4 Results of the pragmatic matters analysis

Results of the pragmatic matters, which are common to all COSTOC courses, are presented in four groups, namely, hardware requirements, software requirements, human resources, and teachers' and students' attitudes and readiness. Since we did not organize an experimentation with teachers and students, we were not able to investigate these pragmatic matters in a real situation.

Hardware requirements

The hardware requirements are not described anywhere, neither in the demonstration lesson nor in the paper documentation that was available for this study. This is a serious shortcoming.

Software requirements

The required software is not listed anywhere. Instructions for installation and running COSTOC courses are delivered as a text file. With these short and simple instructions, installation and execution will succeed.

Human resources

Human resources are needed for installing and introducing tested courses to students. In this case, both can be done by teachers themselves since, as teachers of computer science, they are used to handle computers.

Teachers' and students' attitudes and readiness

Based on our previous experiments (Kopponen et al., 1991), both teachers and students are willing to use COSTOC courses as an extra material to support traditional instruction. Readiness to use computers on computer science courses is not a problem. In any event, the guided tour of COSTOC courses is necessary in order to get familiar with the user interface.

5.5 Computation and Automata: Description and results of the analysis

5.5.1 Course description

Computation and Automata is a full course (i.e., it consists of 11 lessons: introductory lesson 0 + 10 teaching lessons) which presents briefly some of the fundamental topics of theoretical computer science, namely, automata, languages, computability, and complexity.

In addition, it discusses some special topics, such as L systems, Petri nets, and grammar forms. The logical dependency of lessons is represented in Figure 5.3.

According to Salomaa and Maurer (1989), the basic question in the theory of computing can be formulated in many ways: What is computable? For which problems can we construct effective mechanical procedures that solve every instance of the problem? Which problems possess algorithms for their solution? During the 1930's the fundamental development in mathematical logic showed the existence of unsolvable problems. No algorithm can possibly exist for the solution of such problems. For establishing unsolvability, it is essential to have a model of computation.

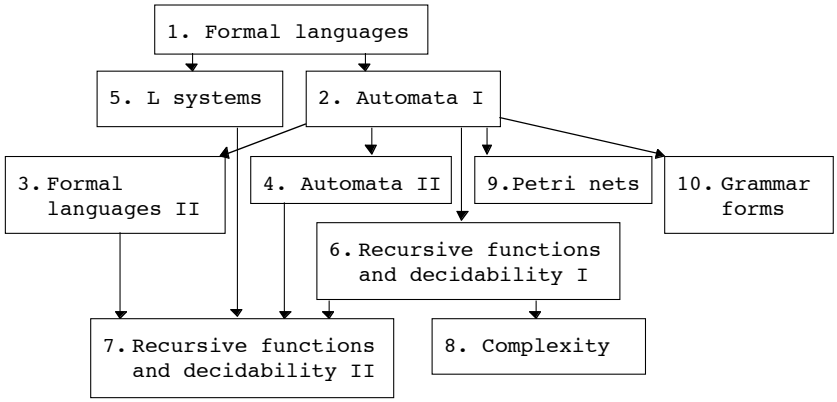

Figure 5.3: Logical dependency graph of Computation and Automata (Salomaa and Maurer, 1989).

This course discusses both general models of computation, such as a Turing machine (see Figure 5.4), grammars, and L systems, and specific models of computation, such as finite automaton. As Salomaa and Maurer (1989) summarize, a good model represents a well-balanced abstraction of a real practical situation - not too far from and not too close to the real thing.

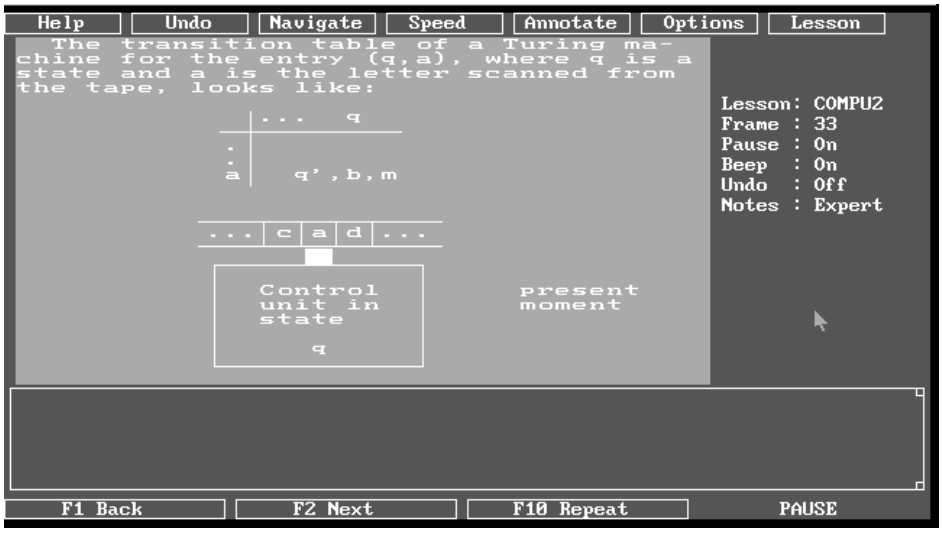

Figure 5.4: Turing machine as an example of the contents of Computation and Automata course.

5.5.2 Numerical results

By classifying frames into navigation, information and question frames and calculating their ratio to the total amount of frames the structures of the tested courses may be depicted. A more detailed classification of information frames into classes, in which frames have properties, such as content types, illustration of information frames, motivation, repetition, recall, and thinking points, forms the course profile. Note that an information frame may include more than one classification property. Thus, the percentages of course profile describe only the existence density of certain property in the information frames. The sturcture and the course profile are presented for all tested courses.

The Computation and Automata course consists of 420 frames. The course profile is presented in Figure 5.5 as a bar diagram and the general structure of this course is presented in Figure 5.6 as a pie diagram.

5.5.2.1 Course contents

Content types of information frames

Computation and Automata course contains 324 information frames that consist of analysis, definitions, equations, productions, programming language code and proofs. Analysis is found in 117 information frames (36%). There exist 133 information frames (41%) containing definitions, which play an important role in this course. Equations are found in 89 information frames (27%), that is, approximately one in every four frames. Productions are found in 104 information frames (32%). Programming language code occurs only in one information frame (0.3%) that presents some examples of languages and alphabets. Proofs are found in 32 information frames (10%).

Illustration of information frames

The information frames are illustrated using examples, pictures and animation. There are 170 information frames (53%) that contain examples. Pictures are used approximately one in every five information frames, that is, there are 63 pictures (20%). There are 83 animations (26%).

5.5.2.2 Instructional support

Motivation, repetition, recall, and thinking points

Learning support of this course consists of motivation, recall and thinking points. Motivation is used in 34 information frames (11%) and recall is used in 35 information frames (11%). Thinking points are rare; there are only 4 information frames (1%) in which they are used.

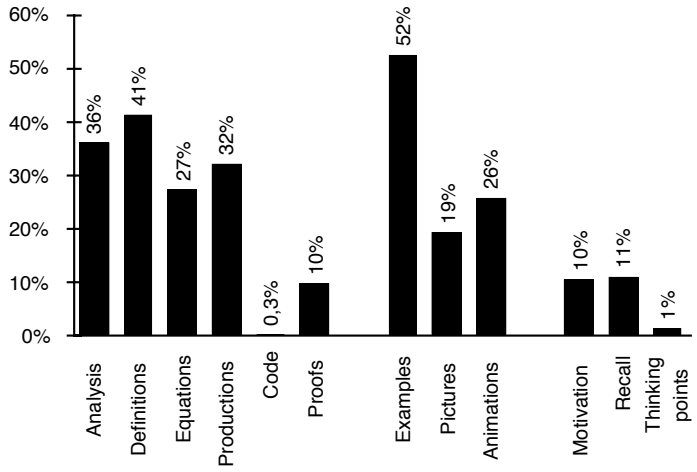

Figure 5.5: The course profile of Computation and Automata.

Structure of the course

Computation and Automata course consists of 420 frames out of which 23 frames (23/420*100%≈5,5%) are navigation frames, 324 (77,1%) are information frames, and 73 (17,4%) are question frames. In addition to the navigation frames, there exist navigation points in 48 information frames. Thus, there exist 71 frames with a possibility to navigate. The frame percentages of Computation and Automata course are shown in Figure 5.6. From all question frames there are 73 questions out of which 67 (92%) are multiple choice questions, 2 (3%) numerical questions and 4 (5%) textual questions.

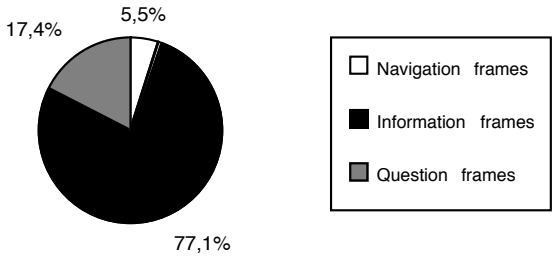

Figure 5.6: Frame percentages of Computation and Automata.

5.5.3 Verbal results and conclusions on Computation and Automata

Course contents

Content types of information frames. The reason for the large amount of analysis is the need to explain abstract phenomena that otherwise might be difficult to understand. The authors have succeeded very well in this task.

The frequency of definitions, that is, two definitions per five information frames, indicates that there are enough frames left to explain and demonstrate new concepts and phenomena. Furthermore, analysis revealed that definitions are distributed to separate lessons quite evenly, thus, students do not get too much new information at a time.

There are differences on frequencies of equations between lessons, that is, the relative amounts vary between 7% and 55%. The lowest frequency is found in a lesson which introduced how to model parallelism by Petri nets. The highest one is found in a lesson which widens and deepens the lesson concerning recursive functions and decidability. Even though the topic is mathematical, most of the equations are very simple. Therefore, students need just a little knowledge of mathematics. This can be explained with the subjects that are discussed in each lesson. Most of them concentrate on the fundamentals of computation theory in which equations are required to a certain extent.

Productions or rewriting rules are the speciality of Computation and Automata course. They are necessary for defining rewriting systems that be used to define languages, which are one of the main topics in this course. The amount of programming code is quite small. This is natural since code is not an essential part of this course.

The large amount of proofs can be explained with the theoretical nature of the course. Within the tested courses, Computation and Automata is the only course in which proofs have a significant role. Some of the proofs are informal, that is, the proofs are verbal and there are examples illustrating them. This kind of style makes them easy to understand, though the authors also give references to formal proofs.

Illustration of information frames. The examples are both verbal and visual (i.e., pictures or animation) and support understanding of the topic very well. Pictures vary from very simple ones to complex graphics that illustrate well, for example, the functions of automata. Animation is used more often than pictures. Animation, like pictures, vary from simple blinking arrows to simulations presenting, for example, how Petri nets are applied in problem solving.

Relevancy of the course contents to the aims. Since the learning aims are not stated clearly, it is difficult to say if the course contents are relevant to the aims. In any event, the quality of the contents is high.

Instructional support

Motivation. Most often motivation is verbal though there are a few motivating examples implemented as animations (e.g., Sieve of Eratosthenes; lesson Compu8, frame 3). Typically, motivation arises by explaining, for example, why something is more useful than something

else. Appearance of motivation is uneven, that is, in a couple of lessons there exists only one motivation point whereas the other lessons may have up to seven motivation points. The lack of motivation points may diminish students' interest in the course.

General view of the course contents. Contents are described both in paper documents and in the beginning of the introductory lesson (i.e., Compu0).

Prerequisites. The introductory lesson of this course informs that no previous knowledge of the subject is needed, though it is good to be 'mathematically minded.' In paper documents, there is no mention about prerequisites.

Informing of the learning aims. The documentation of this course does not state clearly the learning aims.

Structure of the course. The percentage of navigation frames (5,5%) is very low but since there exist many navigation points the real percentage of navigation possibility (17%) is good. Students have enough freedom to choose and not too much to loose the leading idea of the course.

The proportional amount of information frames (77,1%) is very large. Nevertheless, this is not a problem since about every seven information frames contain a navigation point that could have been an independent navigation frame and, in such case, the relative amount of information frames would have been lower (69%). This means that even though the amount of information frames is high, students have enough possibilities to control what to do.

Students have two tries to answer a question and if they fail, the correct answer is shown. The same question is asked until the correct answer is given. The frequency of questions is good since there is one question after approximately four information frames. This means that students' learning is tested quite often and misunderstandings are noticed quickly.

Guidance to use the course. Content-dependent guidance is given both in paper documents and in the introductory lesson, that is, the possible uses of the course are stated.

Interaction. Questions are the only means of real interaction in this course. Thinking points were not found which is a defect. At least some thinking points should have been used, since use of them might enhance students' learning process.

Instructional feedback. Students are given instructional feedback only within question frames. Technical feedback (help) is given when students ask for it.

Repetition and recall. These can be found in every lesson evenly, which indicates that the authors of this course are aware of their importance.

Practicing new skills or knowledge. The only form of practice is questions.

Evaluation of learning outcomes. There does not exist any evaluation of learning outcomes.

5.6 Cryptography and Data Security: Description and results of the analysis

5.6.1 Course description

Cryptography and Data Security is the only half course in this analysis. It represents the basics of cryptography leaving outside certain parts of the theory, such as information theory, in order to avoid many complications, as Salomaa and Maurer (1988) put it. Cryptography and Data Security course discusses security issues which belong to areas where cryptography is a reasonably useful tool, such as digital signatures in electronic mail messages and the secrecy of contents of messages. The logical dependency of lessons is represented in Figure 5.7.

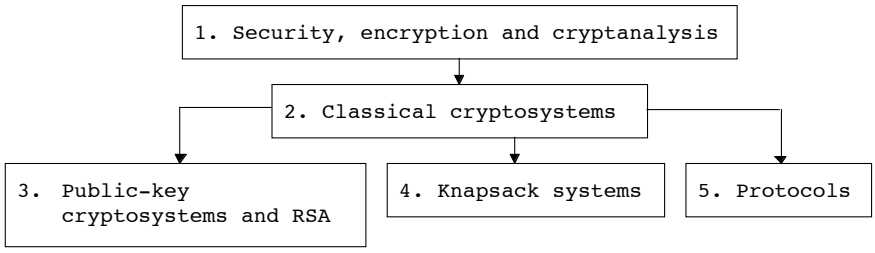

Figure 5.7: Logical dependency of Cryptography and Data Security.

Since the last three lessons are independent, any of them can be studied after lessons one and two. A sample of lesson 1 of Cryptography and Data Security course is presented in Figure 5.8.

Figure 5.8: A snapshot from Cryptography and Data Security course.

5.6.2 Numerical results

Cryptography and Data Security course includes only 206 frames because it is a half course. The classification of information frames leads to the course profile presented in Figure 5.9. The structural appearance of this course is illustrated in Figure 5.10.

5.6.2.1 Course contents

Content types of information frames

Cryptography and Data Security course is built using only the basic content types. It contains 134 information frames which consist of analysis, definitions and equations. Plenty of analysis is used in this course since analysis is found in 22 information frames (16%). Like analysis, definitions have an important role since there exist 58 information frames (43%) containing

definitions. Equations are found in 52 information frames (39%), that is, approximately two in every five frames.

Illustration of information frames

The information frames are illustrated using examples, pictures and animation. There are 92 information frames (69%) that contain examples. There are 28 pictures in 134 information frames (21%); consequently, pictures are used approximately one in every five information frames. Animation is used more often than pictures. There are 51 animations in information frames (38%).

5.6.2.2 Instructional support

Motivation, repetition, recall, and thinking points

Motivation, recall and thinking points are used for supporting the learning process. Motivation is used in 22 information frames (16%). In this course, recall is used in 12 information frames (9%) which is a reasonable amount. Thinking points are rare; there are only 6 information frames (4%) in which thinking points are used.

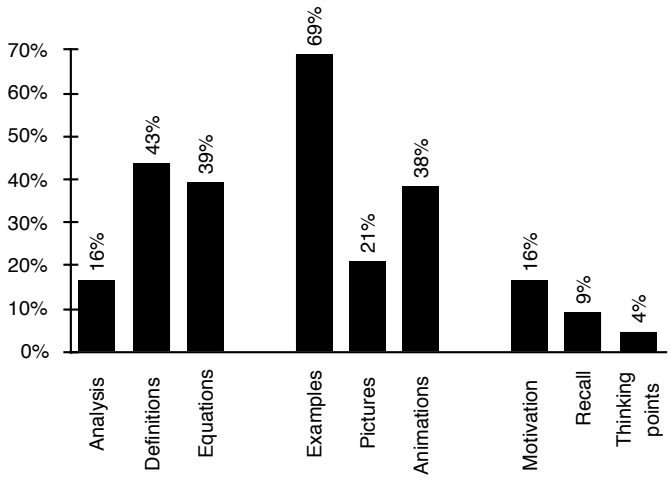

Figure 5.9: The course profile of Cryptography and Data Security.

Structure of the course

Cryptography and Data Security course is a half course and consists of 207 frames out of which 47 (22,7%) are navigation frames, 134 (64,7%) are information frames, and 26 (12,6%) are question frames. In addition to the navigation frames, there exist navigation points in 2 information frames, thus, there exists a possibility to navigate in 49 frames. The 24 question frames contain 26 questions which are all multiple choice questions. The frame percentages of Cryptography and Data Security course are shown in Figure 5.10.

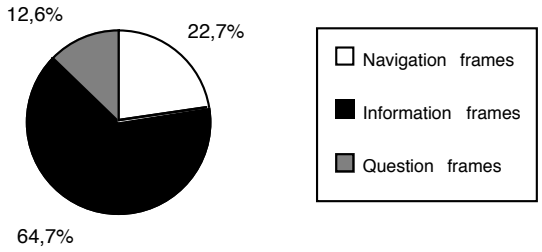

Figure 5.10: Frame percentages in Cryptography and Data Security.

5.6.3 Verbal results and conclusions on Cryptography and Data Security

Course contents

Content types of information frames. The amount of analysis is very good and it indicates that authors have put considerable effort into explaining things in order to deepen students knowledge on certain topics. In some cases, the analysis is illustrated using pictures or animation.

Many definitions are needed because there exist several concepts and methods that are required to understand the subject. The course starts with historical cryptosystems and ends with the latest methods used for encryption and decryption. Thus, it provides students an extensive presentation why cryptography is needed and how it has developed through centuries.

The relatively high amount of equations is natural since cryptography is strongly based on mathematics. Students need to know the fundamentals of mathematics, for example, how to multiply matrices.

Illustration of information frames. The examples are both verbal and visual (i.e., pictures or animation) and support understanding of the topic very well because examples are presented by coordinating them with simple familiar things, such as a telephone directory.

Pictures, which are most often very simple, are used effectively to lighten from time to time difficult concepts and phenomena.

Animation varies from simple blinking arrows to simulations presenting, for example, how messages have been transported in the early days, or how public-key cryptosystems work. Due to programming errors animation does not look like it is probably intended to look in several places, which causes confusion.

Relevancy of the course contents to the aims. Since the learning aims are not stated clearly, it is difficult to say if the course contents are relevant to the aims. In any event, the quality of the contents is high.

Instructional support

Motivation. Most often motivation is verbal though there are a few motivating examples implemented as animations (e.g., what kind of advantages public-key cryptography has; lesson Crypto2, frame 7). The large amount of motivation is good since students are likely to be content when they are explained the grounds and consequences of phenomena. Within this course, motivation is found evenly in every lesson.

General view of the course contents. Course contents are outlined well in the paper documents. In the introductory lesson (i.e., Crypto0), contents are described very briefly.

Prerequisites. Prerequisites are not stated clearly, neither in paper documents nor in the introductory lesson. The level of the course in given in paper documents.

Informing of the learning aims. The documentation of this course does not state clearly the learning aims.

Structure of the course. The proportional amount of navigation frames (22,7%) is quite high but that does not violate the consistency of the course. The relatively high amount of

navigation frames may explain that there exist only two navigation points within information frames. Together these give students enough possibilities to choose what to study.

The percentage of information frames (64,7%) is good. The ratio between information and navigation frames is balanced, that is, there exists one navigation frame per three information frames. This means that information is presented in blocks of reasonable size.

The frequency of questions is fairly good since there is one question after approximately five information frames. In this course, like in Computation and Automata, students have two tries to answer and then the correct answer is shown. The same question is asked once more regardless of whether the student gives the correct answer or not.

Guidance to use the course. There is no content-dependent guidance given, neither in paper documents nor in the introductory lesson.

Interaction. Interaction with students is organised using questions and thinking points. A good thing is that thinking points can be found in four out of five lessons, which indicates that the authors have wanted to stimulate thinking of different topics.

Instructional feedback. Students are given instructional feedback only within question frames. Technical feedback (help) is given when students ask for it.

Repetition and recall. The problem with recall is that it is used only to remind students of the previously taught concepts or methods, and there is no summary at the end of any lesson. Therefore, the overview of this course may remain obscure.

Practicing new skills or knowledge. The only form of practice is questions.

Evaluation of learning outcomes. There does not exist any evaluation of learning outcomes.

5.7 Introduction to Database Systems and the Relational Data Model: Description and results of the analysis

5.7.1 Course description

According to Maurer (1990) the aims of this course are as follows:

1. To present the concepts, notations and languages of databases systems and the relational model in a tutorial fashion,
2. To show the practical applications of the presented knowledge, and
3. To gather enough material to support lecturers teaching classes on database systems and the relational data model.

This course exceptionally consists of twelve lessons (1 introductory and 11 normal lessons), whose logical dependency graph is represented in Figure 5.11. A snapshot of this course contents is represented in Figure 5.12.

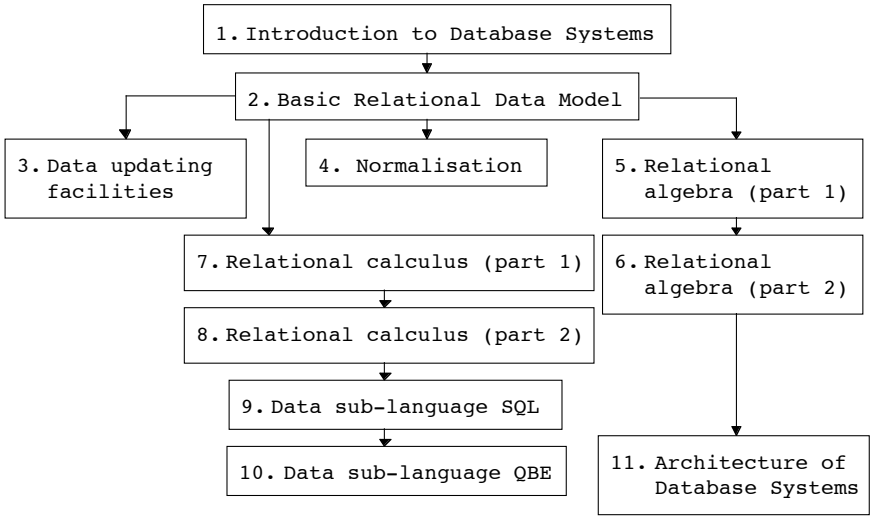

Figure 5.11: Logical dependency graph of Introduction to Databases and the Relational Data Model (Maurer, 1990).

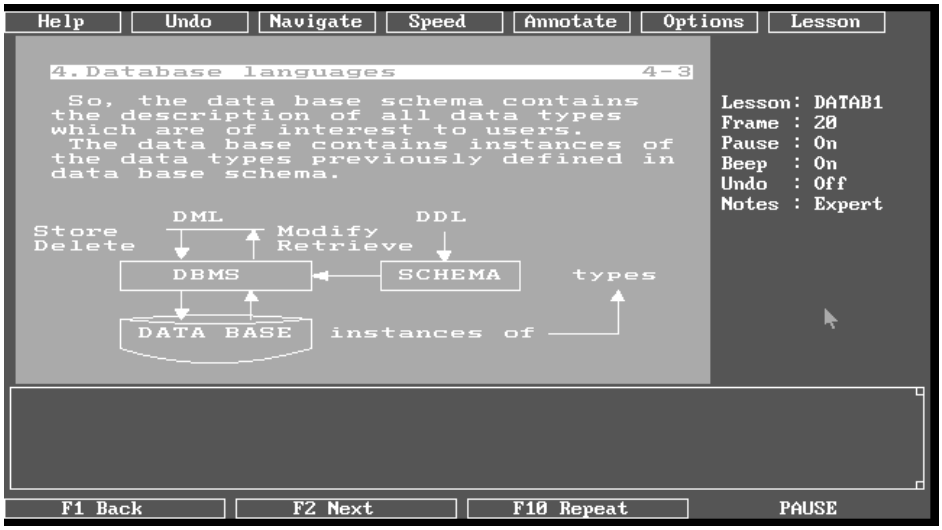

Figure 5.12: A snapshot from Introduction to Databases and the Relational Data Model.

5.7.2 Numerical results

Introduction to Databases and the Relational Data Model course consists of 447 frames. The course profile based on classification of information frames is presented in Figure 5.13 as a bar diagram and the general structure of this course is presented in Figure 5.14 as a pie diagram.

5.7.2.1 Course contents

Content types of information frames

Introduction to Database Systems and the Relational Data Model course contains 244 information frames which consist of analysis, definitions, equations, programming language code, and relations. Analysis is rare because it is found only in 10 information frames (4%). There are 141 information frames (58%) containing definitions. Equations are found in 30 information frames (12%) and programming code occurs in 95 information frames (39%).

Illustration of information frames

The information frames are illustrated using examples, pictures and animation. There are 170 information frames (70%) that contain examples. Pictures are found in 32 information frames (13%), and there exist 126 animations in 244 information frames (52%).

5.7.2.2 Instructional support

Motivation, repetition, recall, and thinking points

As a learning support, only motivation and recall are used in this course. Motivation is used in 13 information frames (5%), and recall is used in 2 information frames (1%).

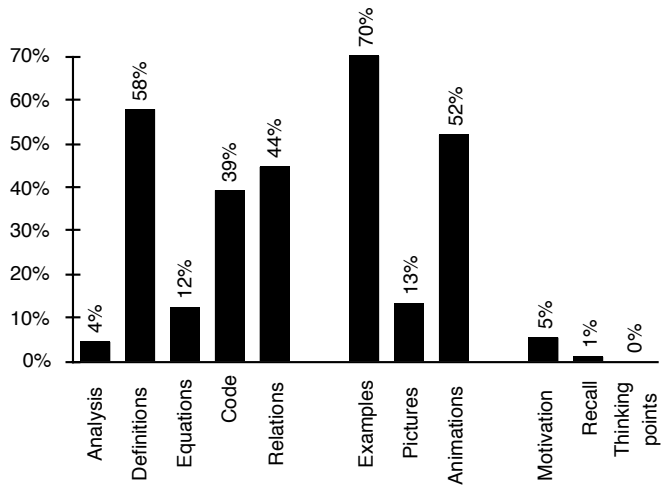

Figure 5.13: The course profile of Introduction to Database Systems and the Relational Data Model.

Structure of the course

This course consists of 447 frames out of which 89 (19,9%) are navigation frames, 244 (54,6%) are information frames, and 114 (25,5%) are question frames. Within this course, there do not exist any additional navigation points. The 114 question frames contain 57 questions (all multiple choice questions) which means that there exist two question frames for each question. The frame percentages of Introduction to Database Systems and the Relational Data Model course are shown in Figure 5.14.

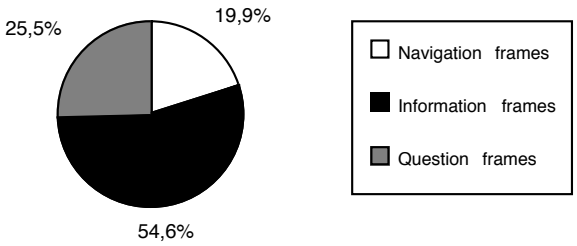

Figure 5.14: Frame percentages in Introduction to Database Systems and the Relational Data Model.

5.7.3 Verbal results and conclusions on Introduction to Database Systems and the Relational Data Model

Course contents

Content types of information frames. The amount of analysis is very low which may cause difficulties for students to understand the topic deeply. In most of the lesson there is no analysis at all. This course is more practical than theoretical and that might be the explanation why analysis is used only seldom.

Definitions plainly have an important role in this course. They are found evenly throughout the lessons which indicates that every lesson provides new information to students.

Formulas are not very typical for this topic, therefore, their amount is reasonable. Equations concentrate on a few lessons which discuss mathematical topics, such as, relational algebra and relational calculus.

Programming language code is an essential part of this course, since databases are processed using special kind of programming languages, namely, Data Description Language (DDL) and Data Manipulation Language (DML). Since the programming samples presented in this course are very simple and the code differs from other programming languages a lot, it is not necessary to know even the basics of programming beforehand.

Illustration of information frames. The examples are mainly visual (i.e., pictures or animation), though there are also verbal ones. Examples are used evenly throughout the lessons and they support understanding of the topic very well.

Most of pictures are used as a part of a clarifying example of concepts or operations. Pictures have a less important role than animation in this course.

Animation is used much more often than pictures. It varies from simple blinking arrows to simulations presenting, for example, how databases are updated using DML. Like pictures, animation is used as a part of examples to assist students to understand different operations.

Relevancy of the course contents to the aims. The contents are of good quality and highly relevant to the aims.

Instructional support

Motivation. Most often motivation is verbal though there are few motivating examples implemented as animations (e.g., why an information model must be dynamic; lesson Datab1, frame 12). The amount of motivation is too low.

General view of the course contents. Both paper documents and the introductory lesson (i.e., DATAB0) contain a detailed description of the course contents.

Prerequisites. Prerequisites are presented very briefly both in paper documents and in the introductory lesson. The level of the course is also given, that is, who the intended users are.

Informing of the learning aims. The aims of this course are clearly stated both in paper documents and in the introductory lesson.

Structure of the course. The relative amount of navigation frames (19,9%) is fairly large. Anyway, since there does not exist any navigation points the amount is reasonable.

The proportional amount of information frames (54,6%) is under the lower bound of the ideal proportion (60%). This indicates that either each information frame contains lots of information and therefore a small amount of information frames are required, or the ratios between navigation, information and questions frames are not balanced. In this case, the latter is true since there exist two question frames per each question and that distorts the ratios. If there were only one question frame per each question, as there usually is, the proportional amount of information frames (63%) would be between the ideal bounds. Therefore, even though the amount of information frames is low the course is in balance.

The amount of question frames differs from the other COSTOC courses noticeable. The reason is that students are allowed to try to answer the question only once and if they fail, one recall frame concerning the topic of the question will be immediately presented to them. These recall frames joined to the questions are not counted in the amount of recall in information frames. The frequency of questions is good since there is one question after approximately four information frames.

Guidance to use the course. Content-dependent guidance is given both in paper documents and in the introductory lesson, that is, the possible uses of the course are stated.

Interaction. Questions are in use but thinking points are missing from this course. It would have been useful to have them since thinking pauses encourage students to reflect the subject even though this topic is more practical than theoretical. This is an obvious defect.

Instructional feedback. Students are given instructional feedback only within question frames. Technical feedback (help) is given when students ask for it.

Repetition and recall. The amount of recall is definitely inadequate. With such little recall students may have difficulties to associate concepts to each other and, since the summaries of each lesson are missing, to get the appropriate overview of the topic.

Practicing new skills or knowledge. The only form of practice is questions.

Evaluation of learning outcomes. There does not exist any evaluation of learning outcomes.

5.8 Sorting Techniques: Description and results of the analysis

5.8.1 Course description

This full course concentrates on presenting major sorting techniques including implementation of the algorithms discussed, the analysis of space and time requirements, and applications. In addition, it introduces the use of various data structures and program design principles, as well as, the importance of efficient sorting techniques and efficient algorithms in general by demonstrating how "a little thinking can save lots of time and effort," as Maurer (1988) puts it. The course also aims to provide enough material on sorting as dealt with in most university classes on data structures and algorithms. The lessons logical dependency graph is represented in Figure 5.15. A snapshot from Sorting Techniques is represented in Figure 5.16.

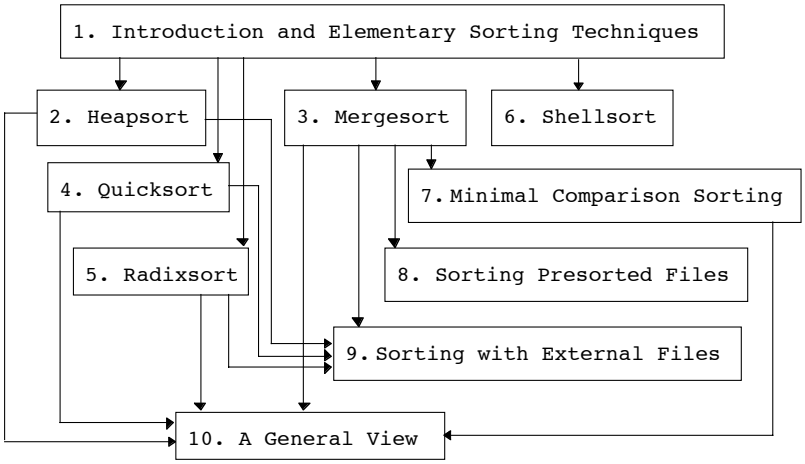

Figure 5.15: Logical dependency graph of Sorting Techniques (Maurer, 1988).

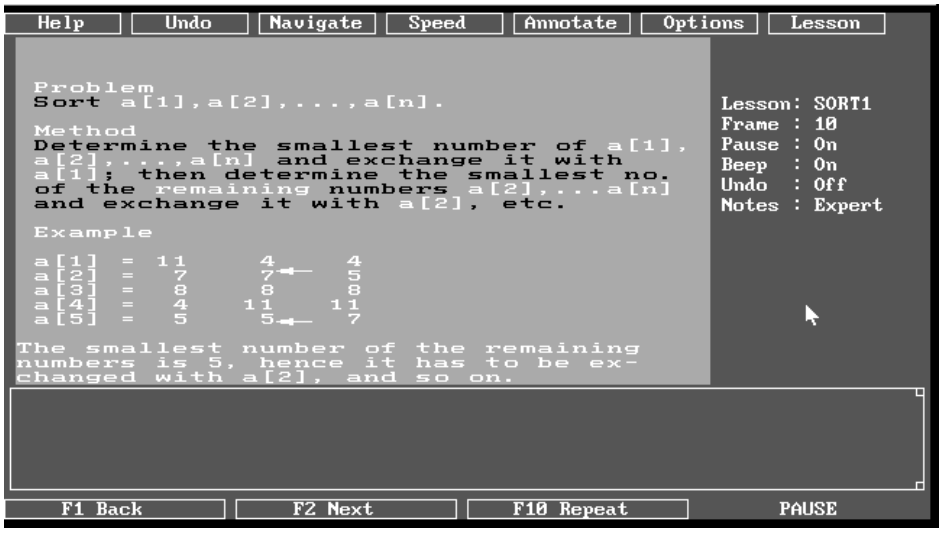

Figure 5.16: A snapshot from Sorting Techniques.

5.8.2 Numerical results

Sorting Techniques course consists of 451 frames. The classification of information frames leads to the course profile presented in Figure 5.17. The general structure of this course is presented in Figure 5.18.

5.8.2.1 Course contents

Content types of information frames

Sorting Techniques course contains 337 information frames which consist of analysis, definitions, equations, programming language code, and proofs. Analysis is found in 125 information frames (37%). There are 56 information frames (17%) containing definitions. Equations are found in 146 information frames (43%). Programming code occurs in 70 information frames (21%) and proofs are found in 11 information frames (3%).

Illustration of information frames

The information frames are illustrated using examples, pictures and animation. There are 163 information frames (48%) that contain examples. Pictures are found in 32 information frames (9%), and there exist 124 animations in 337 information frames (37%).

5.8.2.2 Instructional support

Motivation, repetition, recall, and thinking points

All three forms of learning support are used in this course. Motivation is used in 18 information frames (5%), and recall is used in 30 information frames (9%). Thinking points are rare as there exist only 6 information frames (2%) in which they are used.

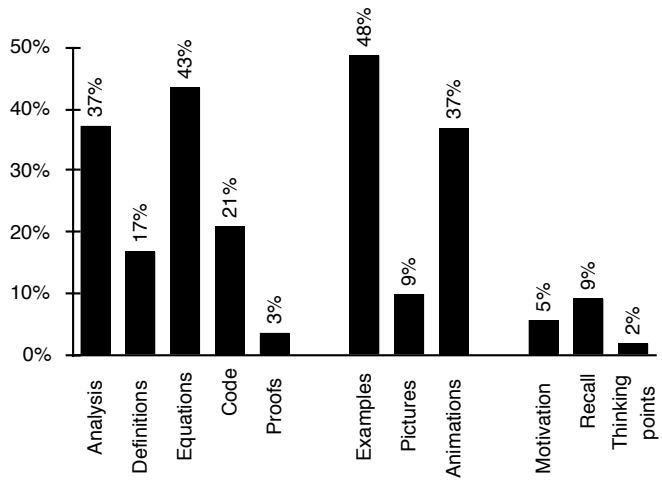

Figure 5.17: The course profile of Sorting Techniques.

Structure of the course

This course consists of 451 frames out of which 71 (15,7%) are navigation frames, 337 (74,7%) are information frames, and 43 (9,5%) are question frames. In addition to the navigation frames, there exist navigation points in 70 information frames. Thus, there exists a possibility to navigate in 141 frames. The 43 question frames include 60 questions out of which 29 (48%) are multiple choice questions, 12 (20%) numerical questions and 19 (32%) textual questions. The frame percentages of Sorting Techniques course are shown in Figure 5.18.

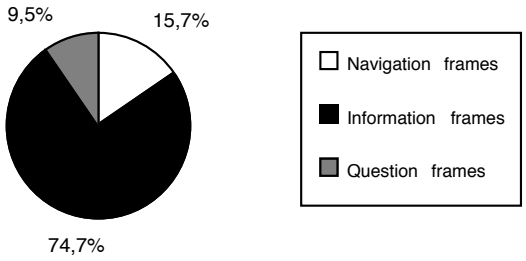

Figure 5.18: Frame percentages in Sorting Techniques.

5.8.3 Verbal results and conclusions on Sorting Techniques

Course contents

Content types of information frames. The exceptionally large amount of analysis is the result of the time and space analysis of sorting algorithms, that is, how much a certain algorithm needs time to finish the task and how much memory is needed. In addition, there are verbal explanations.

The amount of definitions is low since different sorting techniques share the same general concepts, which need to be defined only once. Therefore, even though this course introduces six different sorting algorithms the amount of definitions remains small.

Equations are typical for this topic, therefore, their large amount is reasonable. Equations are essential since sorting algorithms are based on comparisons. In addition, some mathematics is needed when analysing the complexity of different algorithms.

Programming language code is an important part of this course, since sorting algorithms are presented using either programming code or pseudo code, which is a mixture of programming code and natural language. Therefore, students need to know the basics of programming as the prerequisites of this course.

There exist short, less than one frame long proofs (e.g., what is the average time requirement of Quicksort; lesson Sort4, frame 44) and long, multi-frame proofs (e.g., A-sort is not RUNS-optimal; lesson Sort8, frames 31-33). Students need to know mathematics in order to understand proofs though the more detailed proofs are mentioned to be found in the literature.

Illustration of information frames. The examples are mainly visual (i.e., pictures or animation), though there are also verbal ones. Examples support understanding of the topic very well. The small number of pictures, which are compensated for by the large usage of animations, does not impair the effectiveness of the course, though.

Animation is used much more often than pictures. Most of the animation is used to present how a sorting algorithm works (e.g., Radix exchange sort; lesson Sort5, frames 10-12, or Shell sort; lesson Sort6, frames 13-19). In addition, animation is used, for example, to illustrate the comparison of the complexity of different algorithms (lesson Sort6, frames 24-29). Animation, which are found in every lesson evenly, liven this course a lot and help students to understand algorithms that otherwise would be difficult to demonstrate.

Relevancy of the course contents to the aims. The contents are of good quality and highly relevant to the aims.

Instructional support

Motivation. There could be more motivation than there exists because in many lessons there is only one motivating phrase or not even that. Within this course motivation is always presented verbally. This is easy to understand since it would be difficult to invent, for example, motivating animation on this topic.

General view of the course contents. Course contents are described in detail in paper documents. Description in introductory lesson (i.e., SORT0) is less detailed.

Prerequisites. Prerequisites are stated clearly both in paper documents and in the introductory lesson. Also the level of intended users is informed.

Informing of the learning aims. The aims of this course are clearly stated both in the paper documents and in the introductory lesson.

Structure of the course. The proportional amount of navigation frames (15,7%) is appropriate and students have enough possibilities to control the course flow.

The percentage of information frames (74,7%) exceeds the upper bound of the ideal percentage (70%). This indicates that there are either too many information frames or too few question frames considering that the amount of navigation frames is ideal. Since it is supposed that this amount of information frames is required to present all information wanted, it is obvious that there are too few question frames. There exists one question frame per eight information frames which is far too little. If the amount of questions is used for counting, the ratio is one question per six information frames which is better but not good enough. Therefore, there should be more questions in order to test students learning properly.

Students are able to try to answer a question twice. If they fail, the correct answer is usually shown but no confirmation is demanded. Questions include multiple choice questions that may have more than one correct choice. In such cases all correct answers are given to students.

Guidance to use the course. Content-dependent guidance is given both in paper documents and in the introductory lesson, that is, the possible uses of the course are stated.

Interaction. In addition to questions, there exist some thinking points. Unfortunately, they are rare and accumulated into four lessons leaving six lesson without any thinking points which is not recommendable. There should be more thinking points to remind students of the importance of their own thinking.

Instructional feedback. Students are given instructional feedback only within question frames. Technical feedback (help) is given when students ask for it.

Repetition and recall. Recall is distributed evenly in lessons and it assists students to connect previous concepts and methods to new ones. At the end of each lesson there is a summary of contents. This is an appropriate way to repeat most important things of each lesson. The last lesson gives a really valuable overview of the whole course.

Practicing new skills or knowledge. The only form of practice is questions.

Evaluation of learning outcomes. There does not exist any evaluation of learning outcomes.

5.9 Systematic Programming: Description and results of the analysis

5.9.1 Course description

This course focuses on teaching students to program in a systematic way. It introduces the basic constructs of a higher programming language, their syntax, their use and their verification. In addition, the main data types are discussed as well as when to use which data structure and why to use it. Finally, the course represents a program development method, a testing method and the rules of verification. The lessons logical dependency graph is presented in Figure 5.19. The lessons are recommended to study in numerical order with the exception that lessons three and four may change their places. Figure 5.20 represents a snapshot from Systematic Programming.

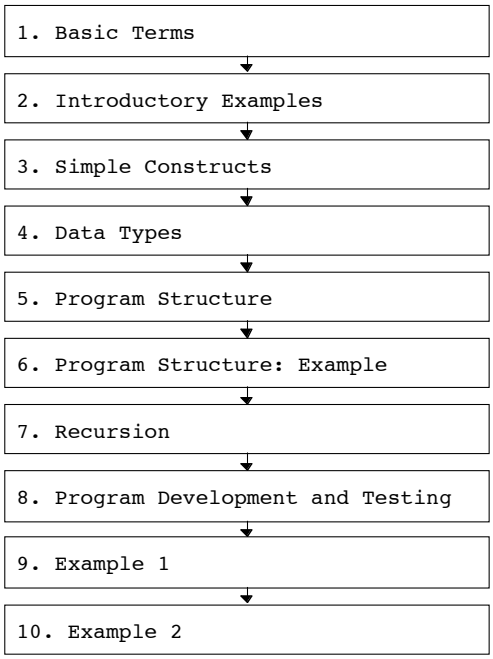

Figure 5.19: Logical dependency graph of Systematic Programming.

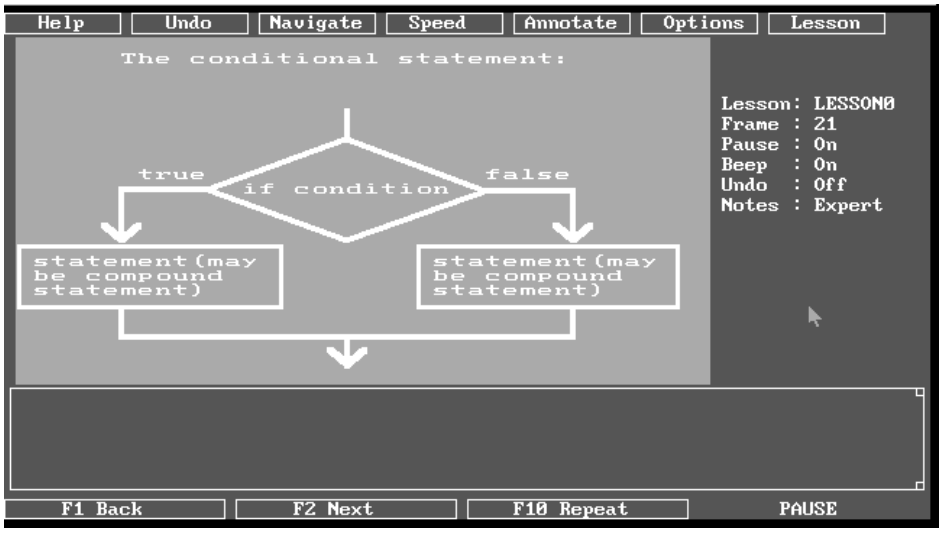

Figure 5.20: A snapshot from Systematic Programming.

5.9.2 Numerical results

Systematic Programming course consists of 475 frames. The classification of information frames leads to the course profile presented in Figure 5.21. The general structure of this course is presented in Figure 5.22.

5.9.2.1 Course contents

Content types of information frames

Systematic Programming course contains 322 information frames which consist of analysis, definitions, equations, and programming language code. Analysis is found in 13 information frames (4%). There are 69 information frames (21%) containing definitions. Equations are found in 91 information frames (28%) and programming code occurs in 173 information frames (54%).

Illustration of information frames

The information frames are illustrated using examples, pictures and animation. There are 269 information frames (84%) that contain examples. Pictures are found in 49 information frames (15%), and there exist 124 animations in 322 information frames (39%).

5.9.2.2 Instructional support

Motivation, repetition, recall, and thinking points

Motivation, recall and thinking points are used for supporting learning in this course. Motivation is used in 40 information frames (12%), whereas recall is found in 21 information frames (7%). There exist only 3 information frames (1%) in which thinking points are used.

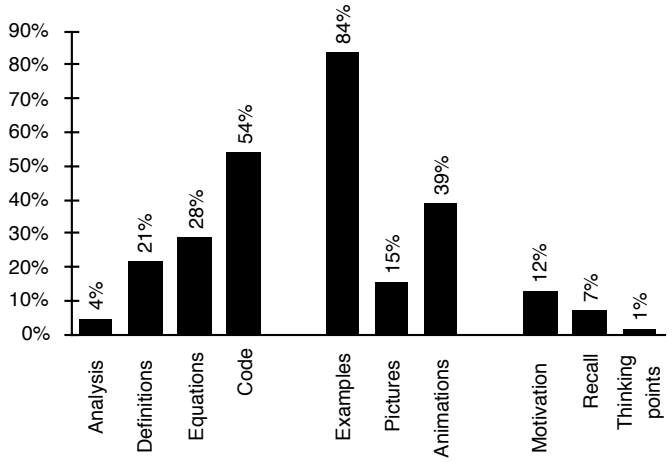

Figure 5.21: The course profile of Systematic Programming.

Structure of the course

Systematic Programming course consists of 474 frames out of which 104 (21,9%) are navigation frames, 322 (67,9%) are information frames, and 48 (10,1%) are question frames. In addition to the navigation frames, there exist navigation points in 21 information frames. Thus, there exists a possibility to navigate in 125 frames. The 48 question frames contain 59 questions out of which 26 (44%) are multiple choice questions, 12 (20%) numerical questions and 21 (36%) textual questions. The frame percentages of Systematic Programming course are shown in Figure 5.22.

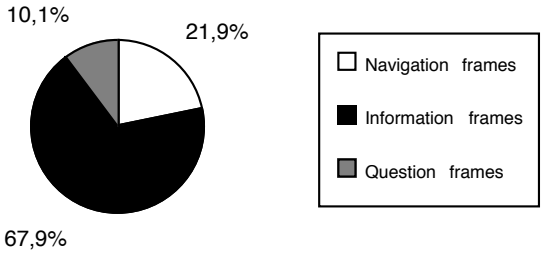

Figure 5.22: Frame percentages in Systematic Programming.

5.9.3 Verbal results and conclusions on Systematic Programming

Course contents

Content types of information frames. The amount of analysis is very low in this course. In six lessons analysis is totally missing which raises suspicions that the contents of those lessons may remain superficial for the students.

The amount of definitions is reasonable because the topic is concise and the number of concepts is limited. Definitions are distributed equally to each lesson. Thus, students get new information in proper portions.

The amount of equations is relatively high because it includes even the simplest equations. Students need to have some mathematical skills since the examples are most often more or less mathematical (e.g., how to develop a program counting x raised to the power y; lesson2, frames 7-29).

Programming language code is an essential part of this course. Students do not need any previous knowledge concerning programming or computers since the course starts from the basics of programming.

Illustration of information frames. There exist two types of examples, namely, short supporting examples and large introductory examples. The first ones are illustrating definitions and are mostly verbal. The large ones consist of multiple frames and contain both text and pictures or animation (e.g., how many bits to represent x; lesson2, frames 31-48). Examples support understanding of the topic very well.

Pictures are used to illustrate, for example, the basic facts about computers or the principles of program verification rules. Pictures are distributed evenly to lessons and they clarify and liven the subject well.

Animation is used effectively to help students to understand how computer programs function. Changing background and text colors and colored arrows highlight the current important statement at the time.

Relevancy of the course contents to the aims. The contents are of good quality and highly relevant to the aims.

Instructional support

Motivation. Most often motivation is verbal though there are few motivating examples implemented as animations (e.g., Fibonacci-numbers; lesson1, frames 3-8). Motivation is found in every lesson through the course. This is good since students learn better when they are motivated.

General view of the course contents. Both paper documents and the introductory lesson (i.e., LESSON0) describe the course contents clearly and in detail.

Prerequisites. Prerequisites are stated both in paper documents and in the introductory lesson. The level of the course is also given, that is, the intended users of the course.

Informing of the learning aims. The aims of this course are clearly stated both in the paper documents and in the introductory lesson.

Structure of the course. The percentage of navigation frames (21,9%) is high. Students have a possibility to navigate in approximately every three information frames when all navigation points are included. This is a quite good ratio because there will not be too much information presented to students at any time.

The relative amount of information frames (67,9%) is appropriate. Like Sorting techniques, this course contains too few question frames and questions since there exists only one question per five information frames. It is not reasonable to expect that testing so much information can be done with so few questions.

Students may try to answer a question once or twice. In seven lessons out of ten the questions are collected in one chapter at the end of the lessons. This kind of lesson structure differs from the other COSTOC courses in which possible questions are placed at the end of each chapter. The frequency of questions is fairly good since there is one question after approximately five information frames.

Guidance to use the course. Content-dependent guidance is given both in paper documents and in the introductory lesson, that is, the possible uses of the course are stated.

Interaction. Both questions and thinking points are used in this course. Thinking points are rare, consequently, their positive effects on learning outcomes remain minor.

Instructional feedback. Students are given instructional feedback only within question frames. Technical feedback (help) is given when students ask for it.

Repetition and recall At the end of each lesson there is a summary. That is good since students get an overview of the contents of every lesson. If there were recalls between lessons students would receive a better overview of the whole course.

Practicing new skills or knowledge. The only form of practice is questions.

Evaluation of learning outcomes. There does not exist any evaluation of learning outcomes.

5.10 Summary of the results

It is natural that COSTOC-courses resemble each other since they are the result of one project. Nevertheless, there exist differences in implementation between these courses. The quality of contents varies between the courses.

Course contents are of high-quality in all tested courses. The three general content types of information frames, namely, analysis, definitions, and equations, are common to all tested COSTOC-courses. The relative amounts of these types vary remarkably which can mainly be explained by the nature of each topic, that is, the amounts of definitions and equations depend on the topic and no strict limits can be given. The amount of analysis, however, is not depending on the topic so clearly. Therefore, there should be more analysis in some of the tested courses. As a consequence of too little analysis, students' learning may remain superficial.

Special content types are needed to characterise the differences between the topics. Four out of five courses contain some of these special content types. If there had been tested courses on other subtopics of computer science, presumably there would have been more special content types.

Illustration of the contents is implemented well. In all examined COSTOC-courses, examples, pictures and animation are used in information frames. Examples, which are used most often, are well-chosen and facilitate understanding the current concept or phenomenon. Examples are taken from everyday life if possible. Animation is used more often than pictures. That is natural in CAI courses since animation is the grand advantage of computers in comparison with the other teaching equipment. In tested courses, animation is often a very simple presentation though there exist a few more complex presentations. Graphics used in animation is not very sophisticated but it is capable to express the essential things.

Relevancy of the contents to stated learning aims is good in three tested courses. In two tested courses the aims were not stated and therefore relevancy could not be evaluated.

Instructional support is inadequate. Amount of motivation is not appropriate in most of the tested courses. Information concerning prerequisites, the level of the course and learning aims is partly defective. Interaction, instructional feedback, repetition, recall, and practice of new skills and knowledge are inadequate in all tested courses. In addition, evaluation of learning outcomes is totally missing from all tested courses.

The general view of the course contents is described well in all tested courses. Structure of the tested courses as well as the guidance to use the tested courses are well described in most of them. Frame types are similar in all COSTOC-courses but the proportions of each frame type vary between the courses. These slight differences are not significant since all proportions are between the acceptable limits. Therefore, the general structure of COSTOC-courses is well-designed and offers a proper environment for learning.

User interface of the tested courses suffers from inadequate interactivity. There is often too much text on the display. Even though visual elements are used fairly often, there could have been more of them. Connections between text and visual elements are mostly clear. Use of colors and highlighting vary a lot between the tested courses. Display building is well implemented.

Pragmatic matters are partly in order. Paper documents are incomplete since all technical guidance is missing. Guidance to install and use the tested courses is given only in a computer format, that is, instructions are found from the demonstration course.

5.11 Conclusions on the results

Authors of COSTOC-courses have had several good ideas to offer and organize learning support, such as, motivating examples, summaries, questions, annotation and thinking points. The problem is that the implementation of these ideas is not successful in all cases. For example, annotation gets forgotten easily since there does not exist any place where students are encouraged to make notes. That property might be used more efficiently if there were expert notes made by a teacher in which, for example, students were given a problem to solve.

Good motivation of students is necessary for learning. Therefore, the amount of motivation in some of these courses was not appropriate. There should exist a short motivating introduction at the beginning of each lesson. A couple of sentences or well-posed examples would be enough.

One deficiently used property is thinking points which seem to be used fortuitously. The idea of thinking points could be improved by using annotate function. Students should be asked to write down their answers or thoughts about the subject they were told to think. Their answers could later be checked either by students themselves or by the teacher. Well-stated questions would direct students' attention to essential things and improve learning results.

Recall is used less than it is advisable in most of the courses. There should exist short resumes within a lesson time to time. In addition, there should be a summary at the end of each lesson and the whole course to combine the essential parts of the lesson and the course, respectively.

In addition to questions, there should be offered more practicing possibilities. Some kind of evaluation of learning outcomes should also be available.

From the technical point of view, the strongest advantage of COSTOC-courses is the user interface whose structure is similar in every course. It is simple and easy to learn and, later on, when studying another COSTOC-course, it is already familiar. Nevertheless, some minor problems may rise if courses are still used in instruction since nowadays students are used to hypertext and hypermedia applications in which links are highlighted using different

colors. In COSTOC-courses colors are used only for highlighting important concepts and there do not exist any links behind them.

One problem with the user interface is its minor interactivity. In addition, there should be less text and more visual elements, and the connections between these display elements should be clearer. Another problem with the user interface is the usage of colors. Colors are fixed elsewhere but not in the main window in which each author may decide what colors are used. As a result, two out of five tested courses are strongly multicolored, that is, colors are used quite carelessly. It would had been better to choose a couple of colors for background and text and make sure that they do not contradict with each other. Highlighting, like use of colors, should be designed with care.

Documentation should be prepared better. Paper documents describe the contents of the tested courses excellently but the technical aspects are totally forgotten.

On the whole, the idea behind COSTOC-courses is excellent. Most of the topics of these courses are still valid though they are implemented nearly ten years ago. The problems are that the user interface functions are not self-evident and that the computer technology for which COSTOC-courses are designed is almost obsolete. If the contents of COSTOC-courses could be transferred to a more modern computer environment they would be good teachingmaterial.

6 Conclusions

Conclusions

A review of scientific journals concerning computers in education revealed that there exist many CAI courses teaching computer science. Roughly one third of the CAI courses that were found from the literature were designed for teaching topics involved with software, especially programming (see Table 2.2 and Table 2.3). The reason for this could be that computer programming is a well-defined, limited domain which is fairly easy to represent in a computer format. Another reason might be that programming languages are useful tools that offer access to the distinctions of computer science.

A sixth part of the CAI applications found in the literature were designed for teaching the theory of computation, which is a positive surprise since there was an expectation that there would exist only few, if any, of such applications. It seems that even though the theory of computation is a very theoretical subject, it is appropriate to be taught using computers. Another reason could be that there exists a need to represent abstract concepts and phenomena in an efficient and illustrative way, for which computers offer good possibilities for skilled CAI authors.

The other topics within computer science seemed to be less popular. However, there exist CAI applications on several subdomains of computer science. Our opinion is that all subdomains of computer science are applicable in a form of CAI as long as the instructional aspects and domain-based aspects are considered.

The collection of human learning theories is vast and, therefore, it was difficult to find out the theories that have been applied to computer-aided instruction. The TIP database (Theory Into Practice; Kearsley, 1996) provided a good basis for further investigations. We found approximately 20 human learning theories, which have been applied or designed for computer-aided instruction or technology-based instruction (see Chapter 3). Four learning theories were taken into closer examination in the form of a case study. As a result, we found

out that both the theories and their applications differ from each other a lot, consequently, there exist different learning theories that are applicable to different domains. Therefore, CAI course authors can choose from a wide collection of human learning theories, the most appropriate ones to be used as a design basis for CAI courses. For this reason, there should exist a CAI authoring tool that would support authors by recommending a certain learning theory, or authors could select their favorite theories from a collection of learning theories. Designing such a collection would require an expert of educational sciences, and it could be implemented as a toolbox to existing authoring tools.

We created criteria in order to evaluate the quality of CAI courses. Our criteria consider domain-based, instructional, user interface, and pragmatic demands (see Chapter 4). As far as we know, this is a unique combination for evaluating CAI courses. Domain-based criteria emphasize the quality of the contents of a CAI course and their relevancy to the learning aims. Instructional criteria emphasize stimulating students' motivation; presenting the general view of the course contents, prerequisites, learning aims, and structure of a course to students; offering guidance to use the course and appropriate interaction including instructional feedback, repetition and recall; and supporting students to practice their new skills or knowledge and to evaluate their learning outcomes. User interface criteria focus on interaction, display elements and connections between them, usage of colors and highlighting, and display-building. Pragmatic criteria concentrate on hardware, software and human resources, and users' attitudes and readiness to use computers.

In order to test our criteria, we modified an analysis method based on them. In addition, we wanted to obtain information of CAI courses on computer science. The analysis method uses two means for evaluating a CAI course, namely, structure counting and verbal evaluation. We have presented checklists for both counting and verbal evaluation (see Section 5.1). This analysis method was used for analysing five CAI courses in the domain of computer science (see Chapter 5). These five courses belong to a collection of CAI courses called COSTOC.

The analysis of COSTOC courses revealed that our criteria work well. It indicated that in tested courses there exist both some shortcomings and benefits. The most serious problem was the inadequate instructional support, though the authors of COSTOC courses have had teaching experience which they have applied while designing these courses. However, no usage of any learning theory was recognized. The most valuable feature was the high quality of the contents in all tested courses.

The analysis was carried out by the author. We believe that the results of the analysis would be mostly the same if the analysis were carried out by some other person. The results of the course contents analysis, the user interface analysis and the pragmatic matters analysis would be the same, whereas there might be differencies in the results of the instructional support analysis. The reason is that other persons might consider, for example, motivational

or analytical sentences in a different way than the author has done. However, we believe that the differences would be minor.

Our criteria should be considered as an example of how to design evaluation criteria, rather than an optimal solution for an evaluation process. The criteria have to be flexible for changes in all perspectives; consequently, criteria must be revised from time to time. In the case of our criteria, revisions are fairly easy to do thanks to the checking lists of each perspective. More work is required if the learning theory behind the instructional criteria is decided to be changed. In such a case, the whole instructional criteria must be updated according to the emphasis of the chosen learning theory.

The importance of the analysis is in its results, which give us information on designing high quality CAI courses on computer science. Conclusions on the results are partly domain-independent and, thus, applicable to other domains. These applicable conclusions include conclusions on the results of the instructional analysis, the user interface analysis, and the pragmatic matters analysis.

First of all, the analysis indicated that the experts of the domain are the best designers of the course contents. The reason is that an expert, who knows the domain well, is the most capable to decide what to include in a CAI course. However, the contents should be implemented by a group of persons who represent diverse perspectives, such as the subject domain, educational sciences, and graphical design. Computer science domain has a strong theoretical basis. Therefore, theory of computer science cannot be forgotten when CAI courses are designed. Theory is necessary for understanding abstract concepts and methods of computer science in depth. Further, these concepts and methods are essential for the design process, that is, when developing new computer applications and equipment. Another important thing to remember is the role of mathematics in the computer science domain. Mathematics is a necessary part in computer science education.

Secondly, we found out that an appropriate learning theory should be applied when designing CAI courses, since the instructional support is likely to be better than it would be when designing without using any learning theory. The next question is which learning theory or theories are appropriate, since both theories and people are very different. One solution might be to combine several learning theories using their most suitable parts while designing CAI courses but that requires expertice of education.

Further, CAI authors should put extra effort into designing appropriate instructional and technical interaction, which play important roles within a CAI course. Carelessly implemented interaction can ruin the whole CAI application.

One possibility to enhance instructional support in CAI courses could be an efficient usage of questions. Questions could be interactive tasks which would guide students to apply their new skills and knowledge. Implementing this kind of questions would demand very sophisticated authoring tools with special properties. When computer science is involved,

these properties include, for example, interpreting both free-form answers and pseudo or programming code, or possibility to write equations, proofs or other mathematical text. There is no need to design a new authoring tool since these special properties could be combined into a toolbox that already existing authoring tools could use as an additional package. Designing the contents of such toolbox requires more research work to be done in future.

Additionally, we found out that CAI authors should pay a lot of attention to the appearance of the user interface. Especially the usage of colors should be considered carefully since poorly chosen colors may distract users' attention to irrelevant things. If the user interface is well-designed, users can forget it and concentrate on the essentials.

Finally, we discovered that pragmatic matters should not be forgotten. It is important to provide users with a proper documentation containing requirements for hardware and software, as well as instructions for implementation and usage.

In recent years possibilities to use computers in everyday life have increased remarkably. Internet and World Wide Web (WWW) are available for almost everyone who is interested in them. Schools are already using these new media for several purposes, such as distance education and information exchange, and it seems that in the future the usage of computers in instruction will increase further. Computers are becoming the basic tools of instruction along with books and other equipment in many domains.

Our opinion is that the best usage of computers is when they are considered similar to other tools used in instruction. Teachers are responsible for designing and organizing lessons. If computers are helpful for these tasks they should be used. Students' motivation is a prerequisite for learning. Therefore, teachers should make sure that students become motivated. This may happen through teachers or through computers. Since computers are restricted with their expressions, it is quite difficult to use them effectively for motivating students. The best combination is teachers guiding students to use diverse tools for learning. These tools include both traditional books and books on a hypertext format, computers and other possible materials, such as experimental equipments.

References

Anderson, J. *Language, Memory and Thought*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.

Anderson, J. *The Architecture of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1983.

Andersson, J. "Skill Acquisition: Compilation of Weak-Method Problem Solutions." *Psychological Review,* **94**(2):192-210, 1987.

Anderson, J. *The Adaptive Character of Thought*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

Anderson, J. *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993.

Anderson, J. and Bower, G. *Human Associative Memory*. Winston, Washington, DC, 1973.

Anderson, J., Boyle, C., Farrell, R. and Reiser, B. "Cognitive Principles in the Design of Computer Tutors." In P. Morris (Ed.), *Modeling Cognition*. John Wiley & Sons, New York, 1987.

Armstrong Laboratory 1996a. *Advanced Instructional Design Associate*. Internet WWW page, at URL: http://www.brooks.af.mil/AL/HR/HRT/HRTD/aida.htm (version current at 18 October 1996).

Armstrong Laboratory 1996b. *Intelligent Performance Support for Instructional Design*. Internet WWW page, at URL: http://www.brooks.af.mil/AL/HR/HRT/HRTD/gaida.htm (version current at 18 October 1996).

Atkinson, R.C., Bower, G. and Crothers, E.J. *An Introduction to Mathematicallearning theory*. John Wiley & Sons, New York, 1965.

Ausubel, D. *The Psychology of Meaningful Verbal Learning*. Grune & Stratton, New York, 1963.

Baek, Y.K. and Layne, B.H. "Color, Graphics and Animation in a Computer-Assisted Learning Tutorial Lesson." *Journal of Computer-Based Instruction,* **15**(4):131-135, 1988.

Baldwin, D. "Three Years' Experience with Gateway Labs." *SIGCSE Bulletin,* **28**:6-7, 1996.

Barnett, B.L. III "An Ethernet Performance Simulator for Undergraduate Networking." *SIGCSE Bulletin,* **25**(1):145-150, 1993.

Barnett, B.L. III "A Visual Simulator for a Simple Machine and Assembly Language." *SIGCSE Bulletin,* **27**(1):233-237, 1995.

Barr, J. and Smith King, L.A. "An Environment for Interpreter-Based Programming Language Projects." *SIGCSE Bulletin,* **27**(1):159-162, 1995.

Barrett, M.L. "A Hypertext Module for Teaching User Interface Design." *SIGCSE Bulletin*, **25**(4):107-111, 1993.

Baxter, N., Hastings, D., Hill, J., Martin, P. and Paul, R. "Introduction to Computer Science: An Interactive Approach Using ISETL." *SIGCSE Bulletin,* **22**(1):31-33, 1990.

Bergin, J., Brodlie, K., Goldweber, M., Jiménez-Peris, R., Khuri, S., Patiño-Martínez, M., McNally, M., Naps, T., Rodger, S. and Wilson, J. "An Overview of Visualization: Its Use and Design." In G. Davies (Ed.), *Proceedings of the First SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education,* Barcelona, Spain, June 2-6, 1996. *SIGCSE Bulletin,* Special Issue, **28**:192-200, 1996.

Berlo, D. *The Process of Communication: An Introduction to Theory and Practice*. Holt, Rinehart and Winston, New York, 1960.

Berque, D., Bogda, J., Fisher, B., Harrison, T. and Rahn, N. "The KLYDE Workbench for Studying Experimental Algorithm Analysis." *SIGCSE Bulletin,* **26**(1):83-87, 1994.

Blythe, S., James, M. and Rodger, S. "LLparse and LRparse: Visual and Interactive Tools for Parsing." In *Proceedings of the Twenty-fifth SIGCSE Technical Symposium on Computer Science Education,* 208-212, 1994.

Boroni, C.M., Eneboe, T.J., Goosey, F.W., Ross, J.A. and Ross, R.J. "Dancing with DYNALAB: Endearing the Science of Computing to Students." In *Proceedings of the Twenty-Seventh SIGCSE Technical Syposium on Computer Science Education,* 135-139, 1996.

Borsook, T.K. and Higginbotham-Wheat, N. "Interactivity: What Is It and What It Can Do for Computer-Based Instruction?." *Educational Technology,* **31**(10):11-17, 1991.

Boyle, T., Gray, J., Wendl, B. and Davies, M. "Taking the Plunge with CLEM: The Design and Evaluation of a Large Scale CAL System." *Computers and Education*, **22**(1-2):19-26, 1994.

Bransford, J.D. and Cognition and Technology Group at Vanderbilt. "Anchored instruction: Why We Need It and How Technology Can Help." In D. Nix and R. Spiro (Eds.), *Cognition, education and multimedia*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

Brown, J.S., Collins, A. and Duguid, P. "Situated Cognition and the Culture of Learning." *Educational Researcher,* **18**(1):32-41, 1989.

Brown, J.S. and VanLehn, K. "Repair theory: A Generative Theory of Bugs in Procedural Skills." *Cognitive Science,* **4**:379-426, 1980.

Brown, M.H. *Algorithm Animation*, MIT Press, Cambridge, MA, 1988.

Brown, M.H. *Zeus: A System for Algorithm Animation and Multi-view Editing*. Digital SRC Research Report 75, 1992.

Brown, R.A. "A Software Testbed for Advanced Projects in Real-time and Distributed Computing." *SIGCSE Bulletin*, **25**(4):247-250, 1993.

Bruner, J. (1966). *Toward a Theory of Instruction*. Cambridge, Massachusetts: Harvard University Press.

Brusilovsky, P.L. "Intelligent Tutor, Environment and Manual for Introductory Programming." *Educational and Training Technology International*, **29**(1):26-34, 1992.

Cagnat, J.M., Gueraud, V. and Peyrin, J.P. "The Arcade Laboratory: An Environment to Help Teach Algorithms." *SIGCSE Bulletin*, **22**(4):37-41, 1990.

Card, S., Moran, T. and Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

Carroll, J.M. *The Nurnberg Funnel*. MIT Press, Cambridge, Massachusetts, 1990.

Caugherty, D. and Rodger, S. "NPDA: A Tool for Visualizing and Simulating Nondeterministic Pushdown Automata." In N. Dean and G.E. Shannon (Eds.), *Proceedings of the DIMACS Workshop on Computational Support for Discrete Mathematics,* 365-377, Rutgers University, Piscataway, New Jersey, March 12-14, 1994.

Chabay, R.W. and Sherwood, B.A. "A Practical Guide for the Creation of Educational Software." In J.H. Larkin and R.W. Chabay (Eds.), *Computer-Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches,* (pp. 151-186). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1992.

Coe, P.S., Williams, L.M. and Ibbett, R.N. "An Interactive Environment for the Teaching of Computer Architecture." *SIGCSE Bulletin,* **28**:33-35, 1996.

Cognition and Technology Group at Vanderbilt. "Anchored instruction and Situated Cognition Revisted." *Educational Technology,* **33**(3):52-70, 1993.

Cognition and Technology Group at Vanderbilt. "Multimedia environments for enhancing student learning in mathematics." In S. Vosniadou, E. De Corte and H. Mandl (Eds.), *Technology-Based Learning Environments: Psychological and Educational Foundations,* (pp. 167-173). NATO ASI Series, Series F: Computer and Systems Sciences, **137**, Springer-Verlag, Berlin, 1994.

Cowley, B., Scragg, G. and Baldwin, D. "Gateway Laboratories: Integrated, Interactive Learning Modules." *SIGCSE Bulletin,* **25**(1):180-184, 1993.

Cronbach, L. and Snow, R. *Aptitudes and Instructional Methods: A Handbook for Research on Interactions*. Irvington, New York, 1977.

Denning, P.J., Comer, D.E., Gries, D., Mulder, M.C., Tucker, A., Turner, A.J. and Young, P.R. "Computing as a Discipline." *Communications of the ACM,* **32**(1):9-23, 1989.

Ernst, G. and Newell, A. *GPS: A Case Study in Generality and Problem Solving*. Academic Press, New York, 1969.

Estes, W.K. "Toward a Statistical Theory of Learning." *Psychological Review,* **57**:94-107, 1950.

Fagin, B. "Two Years of 'The Digital World': Portable Courseware for Technological Literacy." *SIGCSE Bulletin,* **26**(1):97-101, 1994.

Feldman, M.B. "The Portable Dining Philosophers: A Movable Feast of Concurrency and Software Engineering." *SIGCSE Bulletin,* **24**(1):276-280, 1992.

Finkel, D. and Chandra, S. "NetCp - A Project Environment for an Undergraduate Computer Networks Course." *SIGCSE Bulletin,* **26**(1):174-177, 1994.

Fitzgerald, S. and Place, J. "Teaching Elementary Queueing Theory with a Computer Algebra System." *SIGCSE Bulletin,* **27**(1):350-354, 1995.

Foster, L.A. and Hughes, N.L. "Making Files Real with a Virtual Disk." *SIGCSE Bulletin,* **23**(1):199-204, 1991.

Gagné, R.M. "Military Training and Principles of Learning." *American Psychologist,* **17**:263-276, 1962.

Gagné, R.M. *The Conditions of learning*. Holt, Rinehart and Winston, New York, 1965.

Gagné, R.M. and Briggs, L.J. *Principles of Instructional Design*. Holt, Rinehart and Winston, New York, 1974.

Gagné, R.M. and Driscoll, M. *Essentials of Learning for Instruction (2nd Ed.)*. Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

Gee, R. and McArthur, R. "Some Experiences with CAI and NATAL." *SIGCSE Bulletin*, **23**(4):61-64, 1991.

Gibson, J.J. *The Senses Considered as Perceptual Systems*. Houghton Mifflin, Boston, 1966.

Gick, M.L. and Holyoak, K.J. "Schema Induction and Analogical Transfer." *Cognitive Psychology,* **12**:306-365, 1983.

Gong, R. and Elkerton, J. "Designing Minimal Documentation Using the GOMS Model: A Usability Evaluation of an Engineering Approach." *CHI 90 Proceedings*. Association for Computing Machinery, New York, 1990.

Harlan, R.M. "The Automated Student Advisor: A Large Project for Expert Systems Courses." *SIGCSE Bulletin,* **26**(1):31-35, 1994.

Hartley, S.J. "Experience with the Language SR in an Undergraduate Operating Systems Course." *SIGCSE Bulletin,* **24**(1):176-180, 1992.

Heaney, T. *Learning to Control Democratically: Ethical Questions in Situated Adult Education*. Internet WWW page, at URL: http://nlu.nl.edu/ace/Resources/Documents/AERC95.html (version current at 7 October 1996).

Helttula, E., Hyrskykari, A. and Räihä, K.-J. "Graphical specification of algorithm animations with ALADDIN." In *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, 892-901, Kailua-Kona, Hawaii, January 1989.

Helttula, E., Hyrskykari, A. and Räihä, K.-J. "Principles of ALADDIN and other algorithm animation systems." In T. Ichikawa, E. Jungert and R.F. Korfhage (Eds.),*Visual Languages and Applications*, 175-187, Plenum Press, New York, 1990.

Hull, C. et al. *Mathematico-Deductive Theory of Rote Learning*. Yale University Press, New Haven, New Jersey, 1940.

Ingargiola, G., Hoskin, N., Aiken, R., Dubey, R., Wilson, J., Papalaskari, M.-A., Christensen, M. and Webster, R. "A Repository that Supports Teaching and Cooperation in the Introductory AI Course." *SIGCSE Bulletin,* **26**(1):36-40, 1994.

Jacobson, M.J. *Knowledge Acquisition, Cognitive Flexibility, and the Instructional Applications of Hypertext: A Comparison of Contrasting Designs for Computer-Enhanced Learning Environments*. Doctoral dissertation. University of Illinois, Champaign, Illinois, 1990.

Jonassen, D.H. "Hypertext as Instructional Design." *Educational Technology, Research and Development,* **39**(3):5-14, 1991.

Jonassen, D.H. "Cognitive Flexibility Theory and Its Implications for Designing CBI." In S. Dijkstra, H.P.M. Krammer and J.J.G. van Merriënboer (Eds.), *Instructional Models in Computer-Based Learning Environments,* (pp. 385-403). NATO ASI Series, Series F: Computer and Systems Sciences, **104**, Springer-Verlag, Berlin, 1992.

Jonassen, D.H., Hannum, W.H. and Tessmer, M. *A Handbook of Task Analysis Procedures*. Prager, New York, 1989.

Kearsley, G. *Explorations in Learning and Instruction: The Theory Into Practice Database*. Internet WWW page, at URL: http://gwis2.circ.gwu.edu/~kearsley/ (version current at 31 January 1996).

Khuri, S. and Williams, J. "Neuralis: An Artificial Neural Network Package." *SIGCSE Bulletin,* **28**:25-27, 1996.

Kiser, L. "Spatial-visual Ability: Can Computer Visualization Facilitate Achievement?" *Educational Technology*, **27**(5):36-40, 1987.

Knowles, M. *Self-Directed Learning*. Follet, Chicago, 1975.

Knowles, M. *Andragogy in action*. Jossey-Bass, San Francisco, 1984.

Kopponen, M., Kasurinen, V. and Linna, M. "Experimentation of COSTOC-programs." In *Proceedings of the Nordic Conference on Computer Aided Higher Education,* 246-253, Helsinki University of Technology, Otaniemi, Finland, August 21-23, 1991.

Kotz, D. "A Data-Parallel Programming Library for Education (DAPPLE)." *SIGCSE Bulletin,* **27**(1):76-81, 1995.

Lahtinen, S.-P., Lamminjoki, T., Sutinen, E., Tarhio, J. and Tuovinen, A.-P. "Towards Automated Animation of Algorithms." In N. Thalmann and V. Skala (Eds.), *Proceedings of Fourth International Conference in Central Europe on Computer*

Graphics and Visualization 96, 150-161, University of West Bohemia, Department of Computer Science, 1996.

Laird, J.E., Newell, A. and P.S. Rosenbloom. "Soar: An Architecture for General Intelligence." *Artificial Intelligence,* **33**:1-64, 1987.

Lamminjoki, T., Lilja, S. and Ollikainen, V. *Eliot* (in Finnish). Report C-1995-68, University of Helsinki, Department of Computer Science, 1995.

Lave, J. *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life*. Cambridge University Press, Cambridge, UK, 1988.

Lave, J. and Wenger, E. *Situated learning: Legitimate Peripheral Participation*. Cambridge University Press, Cambridge, UK, 1991.

Lees, B. and Cowie J. "Applying Natural Language Technology to the Learning of Operating Systems Functions." *SIGCSE Bulletin,* **28**:11-13, 1996.

Li, S. *R511 Instructional Technology Foundations: Historical Timeline Project: Programmed Instruction*. Internet WWW page, at URL: http://copper.ucs.indiana.edu/~shali/page1.html (version current at May 2 1996).

Li, Z. and Merrill, M.D. "ID Expert 2.0: Design Theory and Process." *Educational Technology Research and Development,* **39**(2):53-69, 1991.

Liffick, B.W. and Aiken, R. "A Novice Programmer's Support Environment." *SIGCSE Bulletin,* **28**:49-51, 1996.

Lim, B.B.L. and Hunter, R. "DBTool: A Graphical Database Design Tool for an Introductory Database Course." *SIGCSE Bulletin*, **24**(1):24-27, 1992.

Linn, M.C. "How Can Hypermedia Tools Help Teach Programming?." *Learning and Instruction*, **2**(2):119-139, 1992.

LoSacco, M. and Rodger, S. "FLAP: A Tool for Drawing and Simulating Automata." In *ED-MEDIA 93, Proceeding of World Conference on Educational Multimedia and Hypermedia,* 310-317, Orlando, Florida, June 23-26, 1993.

Lovato, M.E. and Kleyn, M.F. "Parser Visualizations for Developing Grammars with Yacc." *SIGCSE Bulletin,* **27**(1):345-349, 1995.

Mager, R. *Preparing Instructional Objectives* (2nd Edition). Lake Publishing Co, Belmont, CA, 1975.

Marsden, P. and O'Connell, M. "MuPMoTT - A Multimedia Based Tool Supporting the Teaching of Process Modelling within a Framework of Structured Systems Analysis." *SIGCSE Bulletin,* **24**:116-118, 1996.

Maurer, H. *Sorting Techniques*. COSTOC Course 7, Course documentation, 1988.

Maurer, H. *Introduction to Database Systems and the Relational Data Model*. COSTOC Course 28, Course documentation, 1990.

Meehan, D., Leonard, J. and Schonfelder, L. "An Intelligent Fortran Adviser for Postgraduates and Reseachers." In *Proceedings of CALISCE 91, International conference on computer aided learning and instruction in science and engineering,* EPFL, Lausanne, Switzerland, September 9-11, 1991.

Merrill, M.D. “Learner Control in Computer Based Learning.” *Computers and Education,* **4**:77-95, 1980.

Merrill, M.D. “Component Display Theory.” In C. Reigeluth (Ed.), *Instructional Design Theories and Models*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

Merrill, M.D. *Instructional Design Theory*. Educational Technology Publications, Englewood Cliffs, New Jersey, 1994.

Merrill, M.D., Li, Z. and Jones, M. “Instructional Transaction Theory: An Introduction.” *Educational Technology,* **31**(6):7-12, 1991.

Merrill, M.D., Riegeluth, C. and Faust, G. “The Instructional Quality Profile: Curriculum Evaluation and Design Tool.” In H. O'Neil (Ed.), *Procedures for Instructional Systems Development*. Academic Press, New York, 1979.

Miller, G.A. “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.” *Psychological Review,* **63**:81-97, 1956.

Neff, N. “A Logic Programming Environment for Teaching Mathematical Concepts of Computer Science.” *SIGCSE Bulletin,* **25**(1):20-24, 1993.

Newell, A. and Simon, H. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

Olson, J.R. and Olson, G.M. “The Growth of Cognitive Modeling in Human Computer Interaction Since GOMS.” *Human Computer Interaction,* **5**:221-265, 1990.

O’Neal, M.B. and Kurtz, B.L. “WATSON: A Modular Software Environment for Introductory Computer Science Education.” *SIGCSE Bulletin,* **27**(1):87-91, 1995.

Osborne, M. “APPGEN: A Tool for Teaching Systems Analysis and Design.” *SIGCSE Bulletin*, **24**(1):259-263, 1992.

Piaget, J. *La Naissance de l'Intelligence Chez l'Enfant*. Delachaux & Niestlé, Neuchâtel, 1936.

Polson, M.C. “Cognitive Theory as a Basis for Instructional Design.” In J. Spector, M.C. Polson and D. Muraida (Eds.), *Automating Instructional Design: Concepts and Issues,* 5-22. Englewood Cliffs, New Jersey: Educational Technology Publications, 1993.

Prince, C., Wainwright, R.L., Schoenefeld, D.A. and Tull, T. “GATutor: A Graphical Tutorial System for Genetic Algorithms.” *SIGCSE Bulletin,* **26**(1):203-207, 1994.

Ravden, S. and Johnson, G. *Evaluating Usability of Human-Computer Interfaces: A Practical Method*. Ellis Horwood Limited, Chichester, 1989.

Repo, S. *Mathematics in the Computer Environment. Constructing the Concept of Derivative by Means of the Computer Algebra Program* (in Finnish). Doctoral thesis, Publications in education, N:o 33, University of Joensuu, 1996.

Richards, T.C. and Fukuzawa, J. “A Checklist for Evaluation of Courseware Authoring Systems.” *Educational Technology,* **29**(10):24-29, 1989.

Rifkin, A. “eText: An Environment for Learning Parallel Programming.” *SIGCSE Bulletin,* **26**(1):281-285, 1994.

Rogers, C.R. *Freedom to Learn*. Columbus, OH: Merrill, 1969.

Rosner, M. and Baj, F. “Portable AI Lab for Teaching Artificial Intelligence.” *Education and Computing*, **8**:347-355, 1993.

Saettler, P. *The Evolution of American Educational Technology*. Libraries Unlimited, Englewood, CO, 1990.

Salomaa, A. and Maurer, H. *Cryptography and Data Security*. COSTOC Course 32, Course documentation, 1988.

Salomaa, A. and Maurer, H. *Computation and Automata*. COSTOC Course 5, Course documentation, 1989.

Salomon, G. *Interactionof Media, Cognition, and Learning*. Jossey-Bass, San Francisco, 1979.

Salomon, G., Perkins, D. and Globerson, T. “Partners in Cognition: Extending Human Intelligence With Intelligent Technologies.” *Educational Researcher,* **20**(4):2-9, 1991.

Sanders, I. and Gopal, H. “AAPT: Algorithm Animator and Programming Toolbox.” *SIGCSE Bulletin*, **23**(4):41-47, 50, 1991.

Schank, R.C. *Conceptual Information Processing*. Elsevier, New York, 1975.

Schank, R.C. *Tell Me a Story: A New Look at Real and Artificial Intelligence*. Simon & Schuster, New York, 1991.

Schank, R.C. and Abelson, R. *Scripts, Plans, Goals, and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

Schweitzer, D. “Designing Interactive Visualization Tools for the Graphics Classroom.” *SIGCSE Bulletin*, **24**(1):299-303, 1992.

Selnow, G.W. “Using Interactive Computer to Communicate Scientific Information.” *Americal Behavioral Scientist,* **32**(2):124-135, 1988.

Shannon, C. and Weaver, W. *The MathematicalTheory of Communication*. University of Illinois Press, Champaign, Illinois, 1949.

Shubin, H., Falck, D. and Johansen, A.G. “Exploring Color in Interface Design.” *Interactions:New Visions of Human-Computer Interaction,* **3**(4):36-48, July + August, 1996.

Silver, J.L. “Using Ada to Specify and Evaluate Projects in a Data Structures Course.” *SIGCSE Bulletin,* **23**(1):337-340, 1991.

Skinner, B.F. *The Behavior of Organisms: An Experimental Analysis*. Appleton-Century-Crofts, New York, 1938.

Sloffer, S. *R511 Instructional Technology Foundations: Historical Timeline Project: Teaching Machines*. Internet WWW page, at URL: http://copper.ucs.indiana.edu/~ssloffer/page1.html (version current at May 2 1996).

Spiro, R.J., Coulson, R.L., Feltovich, P.J. and Anderson, D. "Cognitive Flexibility Theory: Advanced Knowledge Acquisition in Ill-structured Domains." In V. Patel (Ed.), *Proceedings of the 10th Annual Conference of the Cognitive Science Society,* (pp. 375-383). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1988.

Spiro, R.J. and Jehng, J.C. "Cognitive flexibility and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter." In D. Nix and R.J. Spiro (Eds.), *Cognition, education, and multimedia: Explorations in High Technology,* (pp. 163-205). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

Spiro, R.J., Vispoel, W., Schmitz, J., Samarapungavan, A. and Boerger, A. "Knowledge Acquisition for Application: Cognitive Flexibility and Transfer in Complex Content Domains." In B.C. Britton (Ed.), *Executive Control Processes,* (pp. 177-199). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1987.

Stasko, J.T. "Tango: A Framework and System for Algorithm Animation." *IEEE Computer,* **23**(9):27-39, 1990.

Stasko, J.T. "Animating Algorithms with XTANGO." *SIGACT News*, **23**(2):67-71, Spring 1992.

Sticht, T.G. "Applications of the Audread Model to Reading Evaluation and Instruction." In L. Resnick and P. Weaver (Eds.), *Theory and Practiceof Early Reading*. Volume 1. Lawrence Erlbaum, Hillsdale, New Jersey, 1975.

Thorndike, E. *Educational Psychology: The Psychology of Learning*. Teachers College Press, New York, 1913.

Tukiainen, M. and Lempinen, R. *Muokattu Ravdenin ja Johnsonin menetelmä ohjelmiston käytettävyyden arviointiin*. (In Finnish). Report B-1994-1. Department of Computer Science, University of Joensuu, 1994.

Tymann, P. "VNET: A Tool for Teaching Computer Networking to Undergraduate." *SIGCSE Bulletin,* **23**(1):21-24, 1991.

Vygotsky, L.S. *Thought and Language*. MIT Press, Cambridge, Massachusetts, 1962.

Wertheimer, M. *Productive Thinking* (Enlarged Ed.). Harper & Row, New York, 1959.

Young, J.I. and Knezek, G.A. "Authoring Tools." *Computers in Schools,* **6**(3):165-173, 1989.

Dissertations at the Department of Computer Science

RASK, RAIMO. Automating Estimation of Software Size During the Requirements Specification Phase - Application of Albrecht's Function Point Analysis Within Structured Methods. Joensuun yliopiston luonnontieteellisiä julkaisuja 28 - University of Joensuu. Publications in Sciences, 28. 128 p. + appendix. Joensuu, 1992.

AHONEN, JARMO. Modeling Physical Domains for Knowledge Based Systems. Joensuun yliopiston luonnontieteellisiä julkaisuja 33 - University of Joensuu. Publications in Sciences, 33. 127 p. Joensuu, 1995.

KOPPONEN, MARJA. CAI in CS. University of Joensuu, Computer Science, Dissertations 1. 99 p. Joensuu, 1997.