Stellingen

behorende bij het proefschrift

Developing Educational Software
Integrating Disciplines and Media

Charles van der Mast
14 februari 1995

I

Elke discipline houdt er een hiërarchie van prioriteiten op na die een overblijfsel is van haar ontstaansgeschiedenis en overlevingsstrategie. Daardoor kan communicatie tussen disciplines zo moeilijk zijn.
Zie: Brenda Laurel (Ed.) *The Art of Human-Computer Interface Design*, Addison-Wesley, 1992, p. 35.

II

Met het doen opstellen van een handleiding voor het ontwikkelen van educatieve programmatuur, als onderdeel van het Informatica Stimuleringsplan (1984-1988), heeft de Nederlandse overheid een impuls gegeven aan het ontwikkelen van educatieve programmatuur in Nederland, binnen en buiten het onderwijs.
-- dit proefschrift--

III

Programmatuur, in het bijzonder educatieve, wordt pas gemeengoed wanneer deze niet alleen door ingenieurs maar tevens door communicatiespecialisten is ontwikkeld.
Zie: Paul Heckel, *The Elements of Friendly Software Design*, Warner Books, 1984, p. 5.

IV

Het verdient aanbeveling om docenten/leerkrachten weer meester te noemen. Misschien begrijpen zij daarmee beter wat er van hen verwacht wordt.

V

Bij het ontwerpen van een informatiesysteem vindt nauwelijks inbreng van andere disciplines plaats wanneer de projectleider de andere disciplines niet kan overzien. Dit pleit voor het instellen van een speciale multidisciplinaire ontwerpers-opleiding die in deze behoefte voorziet.
-- dit proefschrift--

VI

De onderwijsmarkt vraagt met de regelmaat van 'n klok om "betere" onderwijsmethoden en -producten. Dit is een gevolg van het feit dat instellingen, docenten en studenten aan iets nieuws toe zijn.

VII

Degenen die onderwijs genieten dienen onderwijsmiddelen te (kunnen) zien als genotsmiddelen.
-- dit proefschrift--

VIII

De promovendus dient voor het verdedigen van zijn proef-schrift, ter wille van de duidelijkheid, zijn logopediste te raad-plegen.

IX

De informatiemaatschappij is gedateerd wanneer men door de data de informatie niet meer ziet.

Developing Educational Software

Integrating disciplines and media

Developing Educational Software

Integrating disciplines and media

Het Ontwikkelen van Educatieve Programmatuur

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft
op gezag van de Rector Magnificus, Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie
door het College van Dekanen aangewezen,
op dinsdag 14 februari 1995 om 16.00 uur

door

Charles Albert Paul Gerard VAN DER MAST

natuurkundig ingenieur

geboren te Voorburg

Published and distributed by:
Charles van der Mast
Nobellaan 60
2641 XV Pijnacker

Cover design by Katrien van der Mast and Theo Rooijakkers

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Mast, Charles Albert Paul Gerard van der

Developing educational software : integrating disciplines
and media / Charles Albert Paul Gerard van der Mast. -
[S.1. : s.n.]
Thesis Technische Universiteit Delft. - With index, ref.
ISBN 90-9007776-6
NUGI 851
Subject headings: educational software / multi-media /
information systems.

Cover photographs reprinted with permission of
Rabobank Nederland b.v., Fokker Aircraft b.v., and Zwijsen b.v.

For Diederik, Katrien, and Pepijn

TABLE OF CONTENTS

PREFACE

Development and use of educational software has increased due to demand for in-company training at multinationals and big national companies, to large national programs for formal education in several countries, and to emerging perspectives in the free market for educational software and entertainment products. Sometimes the market for expensive strategic products, e.g. aircraft, telephone systems, military equipment, information systems produces a demand for educational software to train employees to use or maintain the product.

The use of information technology to support teaching, training, learning, entertainment, and education in general emerged several decades ago. Many claims about the relative value of the educational software have been made. Although it has been difficult to prove the advantages of educational software over conventional teaching, training and learning its use has increased anyway and many attempts have been made to develop educational software products for different subjects, in a wide variety of educational settings, and for numerous, different target groups.

Developing educational software started within traditional instructional departments and universities. The required expertise on hardware and software was supplied by individual amateurs. Today, developing educational software is also carried out by information system developers; with the required expertise on subject matter and instruction technology being supplied by individuals considered to be "problem owners" or "consumers". In both extreme cases and all situations between, the main bottle-neck for developing educational software successfully is communication and the co-operation between the different disciplines involved. Moreover, this problem has increased due to the new multimedia features modern computers can support.

As spin-off of the development of a Computer Assisted Instruction system for main- and minicomputers I started a course on educational software for students at Delft in 1979. At first, this subject was considered to be interesting for its technical challenge. Later, the course was used as a means to confront technical students with the phenomena of "real" users and other disciplines. During the period I was project manager of the infrastructure for the Dutch Informatica StimuleringsPlan (INSP) I discovered how difficult developing educational software, and implementing it, is. Back at Delft I tried to apply the lessons learnt to the new discipline of designing user interfaces including new media. It was found that educational software design was a good pioneer discipline for user interface design.

In this thesis a new theory is presented for developing educational software with an emphasis on integrating disciplines and media. The theory is tested using three case studies. The aim of this thesis is to improve the practical organization of the development process in multi-disciplinary teams.

This thesis could not have been written without the help of many persons and organizations. Acknowledgements are due to Henk Sol, first for the way he convinced me to think about writing a thesis in 1989, secondly for the way he reminded me to start in 1990, and thirdly for the constructive way he commented the very first outline of the thesis we discussed in the summer of 1991. In 1992 he helped me determining which part of my job could take a lower priority. Then finally, he induced me to start substantially in the Olympic summer 1992. His constructive comments, first as a manager, and later as a participant in the research helped me to resist the attraction of other interesting things of my work and elsewhere.

I would also like to thank the members of the promotion committee for their constructive, and extensive comments from other disciplines which helped me to improve the thesis substantially during the last month.

This thesis could not have been possible without the hospitality of "problem owners" who made available all the information for the case studies. Marianne Spermon from Stichting Teleac for the interesting PC Privé project that induced the paradigm of TV and film making; Rein van der Weide and Joop Tulleken from Rabobank Nederland for the time they spent explaining all the project documents of the project Basisopleiding Zakelijke Dienstverlening. Tom Mous from Fokker Aircraft and Jan van Beckum from BSO/Origin for the opportunity to study all the documents of the Maintenance Training project and to interview them several times. Maud van den Meiracker from Andersen-ECC and Angela Schelfhout from Zwijsen for their permission to study the project documents and the product of the Woordwijs project. Further, I like to thank all my colleagues and friends who collaborated with me during the INSP, as in this period the basis for the thesis was conceived.

Alfred Bork and Bertrand Ibrahim supplied many interesting technical reports for the survey of methods and tools, from Bernard Sodoyer I received materials of his as yet not enough valued Script Editor. I would like to thank Hans de Graaff for his assistance with teaching and coaching our students last year. I hope Bernard and Hans will finish their thesis soon. I would also like to thank all my students who contributed indirectly and invisible to this thesis during our discussions about the subjects of their masters project. I also want to express my gratitude for the intelligible support of Miranda Aldham-Breary, who taught me the English language using the learning-by-doing method. Our meetings were animated by cats. I would also like to thank an old student friend Koos Petterson in Carrollton (GA) who commented on an early draft of the thesis. I admire his spirit and his mental power to survive my thesis.

Further, I am indebted to many friends who took over my role as I neglected my duties on local boards, committees, and my team accepting my apologies without comprehension of the passion of the researcher. Finally, I like to express my gratitude for the total support Melke gave me using her extraordinary social intelligence to understand how communication between disciplines and people works. Our common love for ballet expresses our attitude to multimedia.

Pijnacker December, 1994 Charles van der Mast

1. EDUCATIONAL SOFTWARE

1.1 Introduction

The development of educational software has always been considered to be extremely difficult, both in the seventies (Hebenstreit, 1977; Kearsley, 1982; Stevens, 1986) and in the nineties (McDermott, 1990) and (Van der Mast and Rantanen, 1992). This difficulty has been reported for many educational settings and subject matters, in formal education and in on the job training (Rushby, 1983). The most essential aspect of educational software is interactivity (Gery, 1987). Today, developing user interface software in general is considered to be difficult for many reasons (Myers, 1994). Myers lists the need for multiprocessing, real time requirements and graceful robustness. Extensive research was carried out to determine how students interact with educational software (Kearsley and Hillelsohn, 1982) and how it could be developed (Eberts, 1988) long before human-computer interaction was recognized as an important area to be researched in its own right.

Two ways of viewing the problems of educational software can be found in the literature, one looks at the developed product and "why and how" it worked (Kulik at al., 1980; Eberts and Brock, 1988; Shlechter, 1988; Bork, 1991), the other looks at developing a product and how it can be designed for a specific user[1] (JCAL, 1994). This means that the first approach looks at the "how did it work" for the end user. The second approach looks at "how should it work". The needs of the user are considered throughout the development process.

A lot of attention was paid in the past to technological problems in general and to tools to support programming in particular (Chambers and Sprecher, 1980; Kearsley, 1982). Recently, more attention has been paid to the pedagogical aspects of designing educational software, because it has become apparent that the standard drill and practice formats and tutorial styles are insufficient to build effective educational software. More complex, didactic strategies, specifically created for the educational and organizational context were needed, see e.g. Klep and Kommers (1989); at the same time it was perceived that the individual learning process is unpredictable and cannot be described deterministically (Moonen, 1990).

In the nineties broad experience with the practical application of educational software has shown that the success of educational software does not only depend on careful development. Within the formal education setting, outside the business community, the focus is on the implementation of educational software (Van der Mast, 1989a; Sheingold and Hadley, 1990; Vonk, 1991; Van der Mast et al., 1991; Moonen and Schoenmaker, 1992), and no longer on the development process itself.

Within the formal education system the individual freedom of the institute and of the teacher to approach the training of any subject in a manner they deem expedient means that the existence of good educational software does not necessary guarantee use. Within the industrial/ business setting the use of educational software is often better achieved. This is a problem of commitment (Gery, 1987). Within the business setting, if educational software is seen as having a value for the company it will be used. In the formal education system you have first to convince the middle user, the teachers and trainers, that there is value to found in using

1. By "user" we mean here the user of the final product, in literature "end user" is often used. In the context of this thesis it can also mean the "user" of a tool for the development of products for an "end user".

educational software and it can be difficult to obtain commitment for its use by the institute or by the individual teacher.

Businesses once they commit to the use of educational software will demand that a product is tailor-made for their circumstances. In the formal education setting products tend to be made for a more general audience, there are a greater number of variables, i.e. level of the institute and ability of user/student, etc. Thus educational software used in a business setting is normally a product of "demand" whilst in the formal education setting the software used, if used, is often a product of "supply", that is a piece of software exists and is then taken up by the institute for its use.

Given the above and some 20 years of experience it now seems to be the right time to investigate the triumphs and failures of educational software and its development. The most interesting question is: does educational software differ fundamentally from software for other domains and, if so, whether this results in different requirements for the methodology and the development processes. One approach to developing educational software is to apply regular software development methodologies and tools. Another emphasizes the special characteristics of educational software and advocates special methodologies and tools. Both approaches have been practised with different degrees of success in many large scale projects, as well as in smaller, individual projects.

Computers, and thus the use of software, are an integral part of almost all professional environments and many home environments today. This has lead to an increasing need for and desire to have co-operation between the various disciplines needed when developing educational software. There is a strong, underlying need to "determine" how educational software is developed and how this process can be improved.

There are a number of fundamental questions that require answers:
- how can we cope with the (often over estimated) expectations of users for the possibilities presented by these "new media"?
- how should a project be managed?
- how should the costs be managed?
- how do you define the contents of an educational software product?
- most importantly, how do you define the roles of all participants (teachers, users, designers, subject matter experts, etc.) in the complete life cycle (Van der Mast, 1989a; Van der Mast, 1990b) of a product?

In general to answer these questions, the trend has been to become more professional in the educational approach taken and to incorporate many of the recent benefits to be obtained from developments in information engineering and software engineering (De Diana, 1988). In the same period the disciplines of information engineering and software engineering have payed more attention to the theoretical and practical aspects of user interface design, user participation and usability (Van der Mast and Rantanen, 1992; Van der Mast, 1990a; Van der Mast, 1990c; Nielsen, 1993). Recently, *learner*-centered design has been presented as a new challenge for user interface design (Soloway et al., 1994). The generation of context-sensitive online help also requires integration of user interface and instructional aspects (Spaans, De Graaff and Van der Mast, 1994)

The relevance of the issues mentioned above can be augmented by questions frequently asked in relation to the practical development of educational software, such as:
- should educational software be developed in-house or by third party or by a combination of both?
- should the same methods for development be used for different context settings in formal

education, on the job training, public information, etc.?
- how should the more modern media be integrated within educational products?
- how should co-operation between disciplines be organized within the development teams?

The main goal of this study was to survey literature and practical experiences and to translate recent good ideas into a usable method and guidelines for professional educational software development.

1.2 Educational software

According to the *information paradigm* (Brussaard and Tas, 1980) each set of coherent dynamic phenomena can be abstracted into a real system (RS) and an information system (IS). The IS controls (not exclusively) the behavior of the RS. Information systems contain and generate models of real systems to control or investigate them. This definition is recursive: at a higher level of abstraction each IS is part of a more comprehensive system. Another, more analytic definition of IS identifies it as always consisting of: hardware, software, data, procedures and *people*. In this definition people work according to well-described procedures or are otherwise trained how to work. At a certain level of abstraction the actors (users) in the control and investigation of a RS are not included within the IS. Educational software can be seen as a system in this sense. What are the IS and RS in this context and what is the place of the student/user?

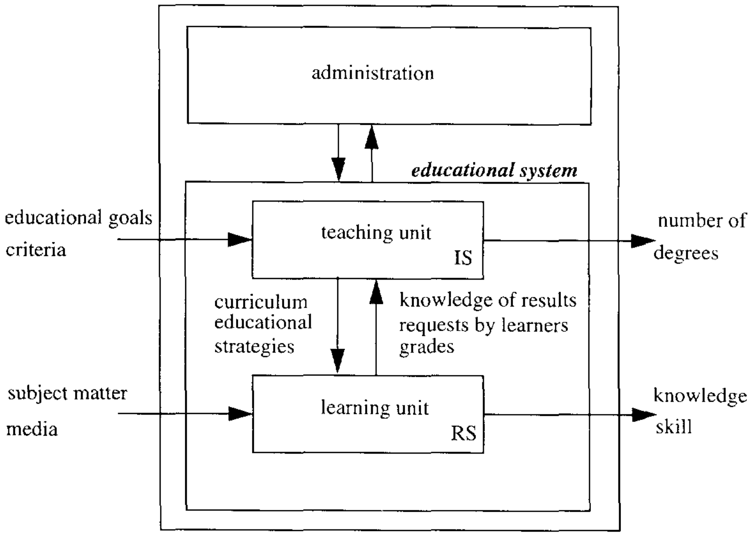


Figure 1.1 Information system (IS) - real system (RS) paradigm applied to a low level within an educational system.

In figure 1.1 an attempt is made to apply the IS-RS paradigm to a lower level within an educational system. Figure 1.1 as a whole can be considered as the RS of a higher level of abstraction. The teaching unit of a school is the IS that controls the RS called the learning unit. The teaching unit consists of the team of teachers and their organizational infrastructure. The learning unit is the RS consisting of individual teachers and students.

The IS sends information to the RS (for example the curriculum and educational strategies) and the IS receives information from the RS (for example grades and other knowledge of results). The environment sends "messages" to the IS (for example educational goals and criteria) and the IS produces the number of degrees issued. The environment sends no messages to the RS but provides materials, services etc. (for example books and educational software) and the RS "produces" graduate students with knowledge and skills. It makes no sense to open the learning unit recursively at this level, since the teaching and learning processes in the RS cannot be further described in this way. The learner has for example, different roles: the student has to be "controlled" by the teacher at some moments and has to be the "operator" of the active learning processes for himself or for another students he communicates with at other times. These and other "double roles" are responsible for a great deal of the complexity of the pedagogical analysis and design (Van der Mast, 1990a; Van der Mast, 1990c).

The student is the most important user of educational software. In traditional didactic styles the role of the student was "designed", sometimes in minute detail. In the modern style of open learning environments the student can choose to some extent, the role she wishes to play. She is free to navigate through and to spend more or less time on components of an educational software package she is interested in and it may be applied as a personal tool to support learning whenever wanted.

The role of the teacher is most important when he has to decide when a student can/must follow a module. The teacher often needs features to adjust the educational software slightly to the technological, organizational and educational context and the environment. The teacher must know the main components of the educational software package and must also know how the student "sees" these main components. This means there should be an opportunity for the teacher to adjust some of the "attributes" of the software concerning paths, scoring, contents and restart facilities. The roles of student and teacher are illustrated in figure 1.2.

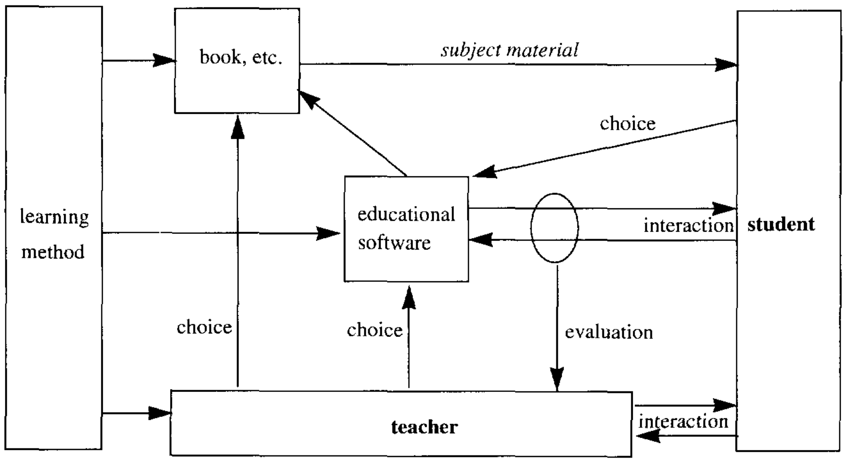


Figure 1.2 Roles within the learning unit of figure 1.1 of student and teacher in the context of educational software (Van der Mast, 1991).

Another argument for the unique character of educational software development can be derived from Giddings' approach.

According to the classification schema for software design given in (Giddings, 1984) educational software can be recognized as the class of software that is distinguished by a strong dependence on the problem statement and the universe of discourse (that is, the class of problem(s) the application is made for). Giddings calls this domain dependency. Giddings recognizes embedded domain dependent systems. "The use of the software may change both the form and the substance of the universe of discourse and, as a result, the nature of the problem being solved. Examples include design automation systems, software engineering systems, office automation systems." Educational software can be added to this list. This special characteristic of educational software will have consequences for the life cycle, see Boehm (1976).

The need for new paradigms for such domain dependent software development is discussed in Agresti (1986). Agresti recognized that software is not homogeneous but that it can be classified based on the relationship of the software to the environment within which it operates. The "environment" can be internal or external to the RS (see figure 1.1). Agresti describes the conventional software life cycle (waterfall) model. The essence of the model is the following sequence of general activities: specification (what), design (how), code and test. "The fundamental weakness of the conventional life cycle model is it's imbalance between analysis and synthesis". Agresti identifies instances of a more flexible development process in literature and shows the characteristics of the type of application involved. Among others he describes two types of applications to which educational software is most similar:

- Functional requirements well understood. User interface not well understood and a significant component of the system.
- Functional requirements not well understood. High interdependence between software and problems to be solved.

Neither type fits well to our implicit definition of educational software. Many other authors have proposed life cycle models for educational software development: Lesgold (1986), Stevens (1986), Ibrahim (1990), McDermott (1990), Bessagnet (1990) and Van der Mast and Rantanen (1992). Van der Mast (1990b) states that the life cycle of educational software should basically be extended by the cycle of *market analysis*. Different roles are involved in this cycle. Entrepreneurs in the market include, commercial publishers for the distribution of the product, schools as funding institutions, individual teachers and last but not least students.

It is very characteristic of educational software that it is extremely user driven e.g. the initial knowledge and the attitude of the students influences the result of using the software. In general, educational software is a "use once" product. Ignorance of the subject matter to be learned is at least partly "destroyed" or changed, learning can never be redone under exactly the same conditions.

Developing educational software may be classified therefore as an ill-structured problem, being a residual concept, indicating what it is or has not: no definite criterion is available to test a proposed solution, much less a mechanizable process to apply the criterion. The problem space is not defined in any meaningful way to encompass all educational, psychological and organizational aspects the designer might consider (Simon, 1973).

Educational software has gone by many names through the years. The name used often depends on the amount of computer control exerted over the instruction (Eberts, 1988). In Computer Managed Instruction (CMI), the computer performs many of the administrative tasks of instruction such as scoring, recording, deciding on the next topic to be taught/learnt and interpreting test results. Instruction material can be presented by off-line media and traditional lectures or instruction groups. In Computer Assisted Instruction (CAI) the computer takes over not only the role of the organizer/scheduler as in CMI but also that of the instructor and assistant. Whole sections of material can be presented by the computer or computer-controlled media while other sections of the material can still be presented by a teacher using traditional media.

The jargon and terminology encountered in the literature is sometimes misleading. Different words are used for the same concepts, often different important points and distinctions are highlighted, e.g. "learning" (Computer Assisted Learning, CAL) denotes more initiative given to the student, while "instruction" (CAI) and "training" (Computer Based Training, CBT) imply that the computer has more control of the instruction process (Rushby, 1983, p. 152). For our purpose, we will not make any distinction between these types (learning, instruction and training) because we are especially interested in the development process and its phasing. It is assumed[1] that the relevant aspect of the development of all styles of educational software is invariant and only differs in the emphasis given to the disciplines involved in the development process. It is also assumed that the differences in the subject matter domain influence the development process in the same order of magnitude as the differences in the types of educational software mentioned here (e.g. the domains mathematics, languages, geography, medical diagnosis and operator training versus tutorials, drill and practice, symbolic and realistic simulations exciting all senses).

Departing from the IS/RS paradigm presented in figure 1.1 we will now define an abstraction level where the user is not part of the information system but is only the operator. At this level the (educational) information system consists of educational software, other educational material (textbooks, video recordings, etc.) and other resources as far as they have to be tailored for use with the educational software being developed. This is part of the "learning unit" in figure 1.1.

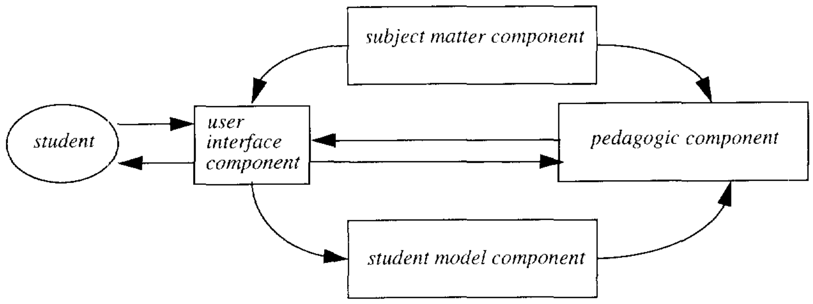


Figure 1.3 Essential components of software forming a part of the educational system operated by the student. When the component for subject matter is missing we have full CMI.

Computer support of the components of educational software given in figure 1.3 is essential. The *interface component* controls communication between student and system. Student input

1. Of course developing simulations may be "more complex" than drill and practise exercises. The identification of training objectives for CBT may also be "more complex" than learning objectives for CAL. We neglect these difference here.

is analyzed and formulated into a machine usable code The output of the system is presented to the student in a suitable form using guidelines for layout and interaction format. The *subject matter component* consists of the set of all facts and procedures belonging to and necessary for the contents to be taught/learnt. The *student model component* represents the model describing the characteristics of the individual student during a session. This model can be very simple consisting of a few variables, e.g. the number of right/wrong responses, the objectives already achieved. The *pedagogical component* contains some simple or complex decision rules to determine the sequence and style of presenting subject matter and feedback to the student, it also determines if the subject is mastered or not. The subject matter component is almost empty for CMI and only refers to traditional resources outside the software system. This subject matter component is worked out in fine details for CAI.

Educational technology and strategy, and subject matter expertise are the premier disciplines needed to develop educational software. With regard to educational software the most important contribution to the learning process is the availability of a great range of interaction techniques and styles during the learning session. Using interactive educational software learning can be motivated in at least three unique and/or alternative ways: the provision of information, the organization and structuring of that information and the physical and organizational learning environment. It should not be forgotten that the very use of educational software itself may well motivate the student to work.

The effect of man-machine interaction[1] (MMI) on software has been studied by several disciplines: e.g. cognitive science, ergonomics, software engineering, etc. Having evolved as isolated disciplines, the increasing availability of commercial graphic user interface (GUI) environments is dissolving the borders and barriers. The development of educational software can be seen as the pioneer discipline in MMI-research and application (Heines, 1984).

Computerized training simulators are a special type of CAI because they consist of a subject matter component extended by an embedded simulator of some part of the real world built explicitly for educational purposes (Hays and Singer, 1988).

In Intelligent Computer Assisted Instruction (ICAI) artificial intelligence techniques are used to perform even more of the instructors tasks. Tasks such as tutoring, natural language question-answering and guidance are performed by the computer. For ICAI especially, the student model component is worked out in great detail. All components show interactive behavior and can be considered as complex processes.

1.2.1 MAIN FUNCTIONS CONCERNING EDUCATIONAL SOFTWARE

When implementing educational software the “run time” situation is important, to study the process of developing it, it is necessary to identify the other main “processes” involved in its use. Educational software has seven main functions, development, presentation, recording, evaluation, distribution, management, guidance/service/support, see figure 1.4.

Development

This is the whole process of problem analysis, educational and functional design, technical specification, programming, testing, documenting, i.e. formulating the objectives, structuring the targets, choosing an instructional strategy, producing and integrating AV materials, editing and formatting the layout, on-line debugging, formal evaluation and producing the final shape

1. The terms human computer interaction or interface(HCI) and computer human interaction or interface (CHI) are other names for man-machine interaction.

in which the materials can be delivered for presentation to students. It is possible to support several of these functions with software tools. This process is the subject of this study.

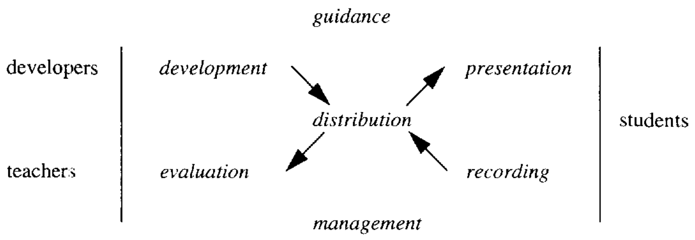


Figure 1.4 The main functions of educational software, cf. Van der Mast (1983).

Presentation

The most important function in educational software is presentation to the user: the student, this should lead to a transfer of information to and from the student; a two-way system of communication is essential. Depending on the instruction strategy chosen, the dialogue is controlled mainly either by the student or by the software. In both cases, presentation is taken to imply the generation and presentation of some instructional material through some device (output), reading of the response of the student, analysis or recognition of that response and initiation of appropriate actions to enable the learning to proceed.

In most educational software, the dialogue with the student is displayed visually on a screen. The visual presentation can be augmented by the use of audiovisual material related to the subject matter. This material may be controlled by the system or manually by the student.

Recording

In many cases, the designers of educational software, or the teacher who prescribes its use, need to have an on-going overview of a student's progress whilst using a piece of educational software. This would include, for example, the responses of all students, some, or of an individual student, their response times in some stages of a dialogue, the number and dates of their logins, their performance on some tests, or the path followed through the material. This is called main function recording and it is necessary for the continuous evaluation of the quality of educational software and the progress of students. As well as this recording of student data for off-line analysis, it may also be desirable to record, at intervals, the status of a student session in case the system goes down.

Evaluation

Evaluation is concerned with what developers have to do after production of the first version or prototype. After *formative* evaluation educational finishing touches can be made. Evaluation is also concerned with measurement and improvement of educational effectiveness in interaction with the students. This type of *summative* evaluation is oriented to analysis of or reporting on the achievement and progress of the students.

Distribution

Distribution of educational software to the students and of recorded data to the developers and the teachers/instructors follows naturally as the next main function. The way this is organized influences not only the costs of the use of educational software but also portability and modifiability and ability to respond to variations in the demand for different versions of educational software packages. In formal education, dissemination of educational software to students on media or via networks is not the only important aspect: distribution to developers and instructors at other educational institutions is also important to share ideas and concepts. The last need is rarely found in commercial environments.

Management
Management of all the previously mentioned functions is also one of the main functions. It includes activities such as administration of all (versions of) lessons and data that should be distributed, and the storage and distribution of all identification codes of students and instructors, each with his own authorization. The management function also includes timetables and scheduling of use of resources, as well as storage of data required to restart a lesson after system failure or a regular end of a session, if such is desired. Management of system security and system performance is also included.

Guidance/service/support
The last main function is the guidance of the students and other users, such as development teams, supervisors, instructors and administrators, by human and on-line help facilities. Although this function need not be done by computer, it will be affected by the features of the system on which the educational software is built. For example it will affect procedures to switch on and off external devices and the possibility of sending e-mail from student to teacher or administrator, an extremely useful property when educational software is used for distance teaching. A part of the guidance function should also be to integrate educational software into a curriculum and provide students with information on how to use the lessons in their individual program.

1.2.2 HISTORICAL BACKGROUND

In the early 1920's educational psychologists[1] began to develop models of the learning process and this has continued to the present day. These models were used as a basis for programmed instruction (PI). PI consisted of text divided into a number of frames each consisting of information to be read, questions to be answered, the correct answer and instructions on how to proceed to the next frame. PI was used on a large scale in the American army during the second world war. Just after the end of the war in the early 1950's the first true "automation" of programmed learning began. Pages of the text to be learnt were turned by mechanical devices (Rushby, 1983). Early computer assisted instruction was based on the computer taking over this mechanical function. As the capacity of computers has grown, so the way in which the computer functions in educational software has changed. Today we have computers that enable us to develop multimedia educational software.

Thus, a natural progression of systems began which contain and control the presentation of interactive educational "frames".

In the USA the application of CAI began with the military services and moved to private investments by and within large companies (e.g. IBM) and universities e.g. Stanford (Suppes

1. See Rushby (1983) for the pioneers: Pressey (1926), Thorndike (1927), Skinner (1961) and Crowder (1963)

and Macken, 1978) and Illinois (Bitzer and Sherwood, 1972).

In Europe development and application were stimulated by (inter)national programs, e.g. UK (NDPCAL, 73-77; Hooper, 1974), the Netherlands (INSP, 1984-1989; Van der Mast et al., 1992), Europe (DELTA program 1990, Cerri and Whiting, 1992, Plomp et al., 1987 and Rushby, 1981). In the Netherlands the Government not only stimulated the production of educational software but it also invested in a technical and organisational infrastructure[1] for its development, see section 3.1. This INSP was followed by the follow-up stimulation programs PRINT, see 2.2.8 and POCO, see chapter 7. In both North America and Europe the development and the use of educational software has become an important activity, especially in the commercial sector.

1.2.3 STYLES IN EDUCATIONAL SYSTEMS

Four major didactic styles can be identified in educational systems (Rushby, 1983). These didactic styles are a result of the concepts, strategies and structures built into the pedagogic component of an educational system, cf. figure 1.3.
1. Drill and practice, presents a programmed sequence of exercises giving feedback as knowledge of result (right or wrong),
2. Tutorial, simulates a teaching dialogue. An item is presented to the student and followed by a question. Depending on the answer the dialogue is continued,
3. Simulation, computer simulation of a scenario executed using variables filled in by the student who can analyse the results of the scenario. The model to run the scenario plays no part in the student's learning process, and
4. Modelling, provides an environment within which students can develop their own models and test their validity within the environment.

In (Rushby, 1983) these styles are ordered from 1-4, the position being determined by the relative degree of control exercised by the teacher and by the student. Modelling shows the highest degree of student control, the medium used is unimportant. Another important distinction made both in the literature and in practice is between education or learning and training. The main reason for this division is the different contexts within which they are used. Training is always close to real activity while education or learning is not closely associated with the student's real world activities. E.g. sex education at school and sex training, see Rushby (1983).

Another major didactic style has emerged as the power of the computer systems available to support graphical user interfaces has increased: the open learning environment. All of the above mentioned didactic styles can be integrated in this "style". An essential "ingredient" is that the learner has the initiative and controls the main steps in learning. Students are invited, in most cases, to use their initiative by the use of metaphor such as the market place or variations on this theme, *Gestalts*. An important advantage of such an open learning system is that the product is one that invites multiple use and can sustain this, it is not a use-once system.

Goal based scenarios (van Aalst, 1995a) and learning by doing (Soloway et al., 1994) need to be added to the above list. Goal based scenarios is a general approach to teaching but it is also viewed as a style in educational software because computers have the ability to offer the environment for this approach using intermediate access to reference media. Plato knew that learning by doing is very effective. When the users job is supported by software then the "doing" is already computerized. Providing instructional software components connected to

1. The author was appointed as the project manager for this part of the INSP.

tasks and actions performed with the application software introduces in such situations learning by doing in a "natural" way.

All styles may be used effectively, the style used will depend on the organization developing/commissioning the product and on the context in which it is used. One general characteristic is that each style will be most suited for some specific set of teaching and learning situations, e.g. learning concepts, training skills, consolidating knowledge.

Educational software systems can also be classified into organizational styles (Chambers and Sprecher, 1980). The first is educational software that supplements the learning situation: *adjunct* CAI. This mostly consists of short programs to support or illustrate concepts which are usually discussed in the regular classroom or that have been taught to the student at an earlier stage.

The second is educational software that substitutes for other modes of instruction: *primary* CAI. These are usually of longer duration and are generally well known and understood in the educational world. This approach is seldom applied for education in schools but is often used for training for modern information-based tasks and jobs. Primary CAI is sometimes used as part of a system for distance learning e.g. Open University. Primary CAI normally consists of modules in which the dominant style is that of simulation.

Another important aspect for classification of educational software is the degree of economic professionalism of the organizational environment where educational software is used. There are considerable differences between using educational software in a commercial organization where the potential students are involved in the production process of the company and in a school setting where achieving "good" quality teaching and learning is the general aim.

It is always necessary for the implementation of educational software to be supported strongly and fully by general and local management. The more professional the organization is, the easier it is to guarantee clear managerial support and commitment from the start-up to the summative evaluation, often a year after the product first goes into use.

When a project fails it is often because the developers have failed to realize the skills and years of experience needed to produce good quality systems. Skills are not learnt overnight, many mistakes will be made before an acceptable result is achieved, see (Sheingold and Hadley, 1990; NCP, 1991; NCP, 1992). It is very necessary for the people involved with developing educational software to be both experienced and committed to producing a good *end* product.

1.2.4 MARKET AND IMPLEMENTATION

Analyzing the complete life cycle of educational software gives rise to the question what actors participate in the life cycle process. The first person is the entrepreneur who is interested in opportunities in the market. The entrepreneur carries out a market analysis and a feasibility study. If he decides to go for a project, he has to start up a project and has to form several teams to do the work up to user testing. Then he or his publisher arrives in the market arena to make a consumer product of the educational software. Next a school may decide to buy the courseware after a proposal made by the teaching team. This decision is made after careful evaluation of several similar products and the costs of introducing and implementing it in the curriculum on the local scale. The individual teacher then has to prepare, schedule and plan the use of the material. The teacher must study e.g. the possibilities for tracking the progress of students. Finally the student will follow the educational software when it becomes necessary for their education. During use errors and ideas for improvement should be registered at the school. The publisher needs to collect these comments to be used when preparing a new release of the educational package.

During this life cycle all the participants mentioned have good knowledge to contribute, but nobody knows the market for educational software as yet. As with other types of automation, in manufacturing, banking, administration, computing, designing, etc., it takes a couple of decades of professional experience on a large scale by both developers and users to reach a state of maturity and know the market.

In a study on successful use of educational software (NCP, 1990) success is defined as "continuing usage". Many factors can be mentioned which are critical to success, such as the expertise of the management of the teaching institute, quality of the product, attitude of the teachers, technical standardization of the local infrastructure. The role of teachers who achieve successful usage of educational software with their students is analyzed, (Sheingold and Hadley, 1990). In the context of this study we are not interested in these factors and roles. Our approach is that registration of sales and scale of use indicates the success of educational software. Of course evaluation of use and interviews of users can provide information for maintenance and improvement of a product. When educational software is developed for a "problem owner" within a commercial organization enough commitment can be expected for the success of the product. This commitment is not yet present today in schools. Therefore it is difficult at the present time to report large scale use of educational software in schools.

1.3 Life cycle of educational software

The successful development of a product is not a guarantee that it is used frequently and economically soundly by any group. Educational software has also to meet the requirements of a real group of users in a cost-effective way. Therefore these requirements must be known and validated in terms of "problems" and "problem owners" of the primary process, i.e. the teachers in formal education and the workers (potential students) in training. This is one of the main problems in automation of human tasks within the information society of today. Many attempts to manage this have been made, see for example how Mantei and Teorey (1988) define development stages in a human factors software life-cycle: market analysis, feasibility study, requirement definition, product acceptance analysis, task analysis, global design, prototype construction, user testing and evaluation, system implementation, product testing, user testing, update and maintenance, product survey. Grudin et al. (1987) present a similar software life cycle with human factor processes. They also discuss the role of the managers at different levels. The roles of marketing as well as development managers in specification and evaluation of the quality of the user interface are worked out in a project schedule.

The model of the life cycle for user and market oriented software given above is more complicated than the conventional waterfall model with isolated phases (Agresti, 1986). The fundamental weakness of the conventional life cycle model is identified as an imbalance of analysis versus synthesis due to the influence of the system approach. Agresti shows the evolution of the conventional model with the changing types of applications and with the changing power of hardware and software. One of his observations is that the life cycle model reflects the time period in which it evolved. Prototyping the user interface, operational specification and transformational implementation are significant steps towards improving the development process. They address the weakness in the conventional life cycle model by providing synthesis in various forms. "Prototyping and operational specification yield behavior-producing objects early in the development process. Transformational implementation also provides for synthesis by automating the generation of concrete code from specifications." Prototyping the user interface seems to be a step that is very applicable to educational software. Agresti (1986, p. 13) presents a list of application types and argues that the types with well understood functional requirements and UI not well understood that are also a significant component of the system (as is the case with educational software), require intensive prototyping as a major activity of developing, operational specification, off-line performance studies, experimentation and project-specific control and integration points. Use of such flexible development process "..will certainly require more management efforts because the control points and intermediate products will vary among projects."

Another development perspective is presented by Henderson (1991). His simple model of development (pictured in the metaphor of a cycle) is that development is composed of five activities: use, observation, analysis, design and implementation, see figure 1.5.

Translated to the domain of educational software "using" is learning by the student, it is the essential grounding of the development process. The product of use is experience. Observation is the interaction with use for the purpose of recording what happens. The product of observation is records. Analysis is searching for regularities in the observational records. The product of analysis is patterns capturing those regularities. Design is the identification and choice of what needs should and will be addressed and the creation of solutions that meet those needs. Design produces specifications, abstract or concrete, expressed in many forms. Implementation is the realizing of the new use situation specified in design. The realization can take the form of an environment as scenarios or scripts. It can be a prototype or it can be a full-scale implementation. Implementation of educational software will require technological but also social and managerial skills. The problem with the metaphor of the cycle is that it does not suggest that the activities can take place concurrently and be interwoven.

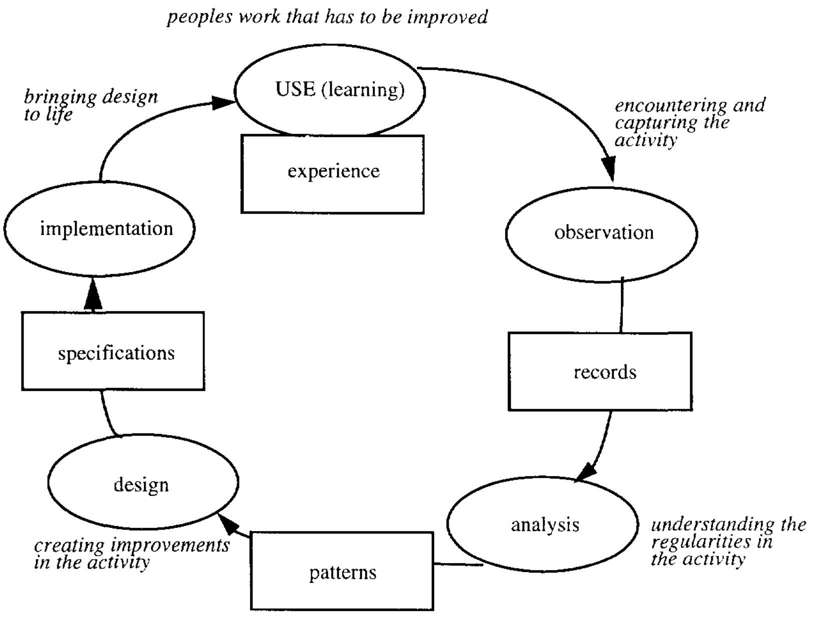

peoples work that has to be improved
USE (learning)
experience
encountering and capturing the activity
observation
records
analysis
understanding the regularities in the activity
patterns
design
creating improvements in the activity
specifications
implementation
bringing design to life

Figure 1.5 Activities and products in the development process (Henderson,

A compilation of the life cycle of educational software mentioned in many reports and papers

(Bork, 1984a; Gery, 1987; Hartemink, 1988; Van der Mast et al., 1991) results in the following list of activities:

problem analysis
  (market analysis if commercial product is aimed)
 feasibility study
   educational
   organizational
   technical
   economical
  global project planning
  problem definition
  product definition
  target group definition
educational design
  context analysis
  subject matter analysis
  didactic strategy design
  media selection
software design: functional design
design of other materials
realization
  technical design and construction
evaluation
  formative evaluation
  field test
  acceptance procedure
production: final product
implementation: in organization or school
maintenance and use

1.4 Strategies for development

Developing educational software can be considered from different viewpoints. It can be seen from the viewpoint of aiming to support e.g. the student, the teacher, the department or unit which needs skilled employees for production or to support independent consumers on the educational and entertainment market. Depending on the viewpoint chosen and the characteristics of the general goals different strategies are needed. The higher and more general the strategy the more comprehensive the necessary methods and tools will have to be. When developing a general strategic product developers have to emphasize the educational context strongly. When developing a professional product for a teacher and/or a student of a particular department or institute developers have to emphasize the individuals need for freedom and flexibility. Some factors are important in each production strategy. Bork (1984) lists the following factors:

1. the emphasis should be on pedagogical aspects, rather than on technological aspects;
2. the best possible teachers should be involved;
3. the production system should allow a very wide range of pedagogical strategies;
4. the full capability of the new media should be used;

5. reasonable and estimable development costs should be assured;
6. insist on extensive evaluation;
7. units must be easily modified;
8. units must be easily portable;
9. large scale curriculum development should be encouraged.

1.5 User interface design

The essential aspect of educational software is its interactivity. So, educational software can be regarded as an area where educational design and user interface design meet. For this reason it is interesting to describe the components, characteristics and techniques used in developing educational software that are inherited from graphical user interface (GUI) and multimedia design.

Many concepts are used to describe man-machine interactions. One comprehensive description is the four-level approach of Foley et al. (Foley and Wallace, 1974), (Foley et al., 1990) who define human computer interaction (HCI) as all input from human to computer, all output from computer to human and the sequencing and ordering of all components involved. Two languages are involved in this interface, one for each direction of information. One language is expressed via actions on input devices. The other can be expressed visually through graphical objects and text, and aurally through tones and synthesized words. All languages have the two components meaning and form, and meaning has two elements: the conceptual and the functional. These four levels are called conceptual, semantic, syntactic and lexical. Foley et al. (1990)(p. 394) define the levels as:

"the *conceptual design* is the definition of the principal application concepts that must be mastered by the user, and is hence also called the user's model of the application. The conceptual design typically defines objects, properties of objects, relationships between objects and operations on objects".

At this level the choice of a metaphor[1] for the user interface is considered. The conceptual level is where the user's mental model of the interactive system is chosen. The characteristics of the users and the "work" they will do using the IS need to be known accurately. The results of a task analysis may be a prerequisite for the conceptual design.

"*Semantic design* (also called functional design) specifies the detailed functionality of the user interface: what information is needed for each operation on an object, what errors can occur, how the errors are handled, and what the results of each operation are. It defines the meanings, but not the sequence of actions or the devices with which the actions are conducted."

The semantic level is where the functionality of the interface is specified: what information is needed for each operation on an object, what errors can occur, how are the errors handled and what are the results of each operation. Meanings, not dialogues are specified.

"*Syntactic design* (also called sequencing design) is part of the form of an interface and defines the ordering of inputs and outputs. For input, the syntactic design comprises the rules by which indivisible units of meaning are formed into complete sentences. Units of meaning cannot be further decomposed without loss of meaning. For output, the notion of sequence

1. The metaphor applied in a user interface should allow perception (by all senses of the user involved in perception) of the objects needed to perform the users task with the functionality offered by the IS. Examples include: command style dialogue, menu style dialogue, desktop with direct manipulation, other Gestalt metaphor of the main concepts of an application domain.

includes spatial and temporal factors. The units of meaning in the output sequence, as in the input sequence, cannot be further decomposed without loss of meaning."

At the syntactic level the ordering of inputs and outputs is defined in time and condition space. It is where the dynamic aspects of the interface are defined. This level is not restricted to the dialogue metaphor. It is also applicable to direct manipulation and event-driven interaction.

"*Lexical design* (also called hardware binding design) is also part of the form of an interface. The binding determines how input and output units of meaning are actually formed from hardware primitives. The input primitives are whatever input devices are available, and the output primitives are the shapes and their attributes provided by the graphics subroutine package."

The lexical level is the level where the form of the interface is defined as a way of binding with the hardware primitives. It is where the static aspects of the interface are defined.

Applied to educational software the conceptual level refers to design metaphors used to represent the subject matter and the didactic strategy. The semantic level refers to the educational modelling of the subject matter to provide an instructional strategy. The syntactic level refers to specification of the dialogue during learning. The lexical level refers to the layout and other static details during the presentation of subject matter and interaction on the screen or with other devices.

Shneiderman (1983) introduced the term "direct manipulation" to describe the class of graphical user interfaces characterized by:
1. continuous representation of the objects and actions of interest;
2. physical actions or labelled button presses instead of complex syntax;
3. rapid incremental reversible operations giving rise to immediately visible impact on an object of interest.

The syntactic-semantic object-action (SSOA) model of user behavior was originated to describe, among other things, direct manipulation, Shneiderman (1992). It is based on a representation of the user's knowledge in long-term memory. Syntactic knowledge is varied, device dependent, acquired by rote memorization, and easily forgotten. Semantic knowledge is separated into the computer and application domain. Within these domains, knowledge is divided into actions and objects. Semantic knowledge is structured, device independent, acquired by meaningful learning and stable in memory. With many styles of user interfaces, users should possess substantial computer-domain semantic knowledge. This knowledge of course consists of artifacts created to support the user-computer dialogue (e.g. key-strokes, command language, error messages, directories, backup files, etc.)

With direct manipulation user interfaces need only a modest amount of computer-related semantic knowledge and syntactic knowledge. The reason is that the necessary computer-related semantic knowledge is decreased by use of the metaphor of direct manipulation. At the same time the need for syntactic knowledge is also decreased.

Direct manipulation is recommended as a general interaction style for educational software to avoid, as much as possible, a student's need for semantic and syntactic knowledge of computer-domain objects and actions.

Two problems that seem new to direct manipulation user-interface are the *illusion of manipulable objects* and *revealed structure*. The first problem is to invent a set of objects that are appropriate to the intended application. The graphic design challenge is to represent them in a convincing way using appropriate metaphors. The second problem is to provide special views, handles or special symbols to reveal structures when necessary.

In IS design, software engineering and database management systems UI design is becoming an important subject of study. Both the separation of UI and application semantics and the integration of both, represented in one data model and one database are the subject of study in major projects. (Versendaal, 1991; Van der Mast and Versendaal, 1993).

The Seeheim model (Green, 1983) was one of the first attempts to separate UI and application during run-time. The main problem with separation is how to support semantic feedback[1]. In some types of educational software semantic feedback is essential (especially when educational software does not have transaction oriented dialogues but generative exercises and simulations). For educational software, separation of user interface and subject matter design may also be an important goal.

1.6 Observations and questions

The general questions mentioned in 1.1 can be elaborated by what has been said in the previous sections. A basic point of all questions is that developing educational software demands close collaboration by rather traditionally different disciplines. The following observations can be made.

1. Subject matter experts, educational specialist and senior teachers perceive it as extremely difficult to collaborate in a team to make a satisfactory specification of the educational problem and a consistent conception of the proposed solution. Some of the reasons for these difficulties may be an inability to recognize and acknowledge mutual responsibilities and skills and a lack of accurate communication. De Diana (1988) ascribes the difficulties to a lack of validated methods, a lack of theoretical and pedagogical theories, a lack of communication and agreement between co-operating disciplines and to technical difficulties during the realization of a product. Van Offenbeek (1993) ascribes such difficulties in conventional information system design partly to poor communication skills within the team. The quality of *communication* between disciplines involved seems to be a problem.
2. Another observation is that to achieve a good design, a great amount of prototyping, reviewing, user participation and formative evaluation is necessary. To some extent developing educational software can be compared to developing a new musical or designing a large multi-purpose building. Intense feeling and *imagination* is demanded from the designers with regard to the design; as well as the quantitative aspect there is a strong qualitative input in the solution that cannot be described deterministically. A comparison with filmcraft can also be made (Heckel, 1991; McKendree and Mateer, 1991; Young and Clanton, 1993). As a consequence of this "qualitative" characteristic of educational software the phase of conceptual and functional design takes a very long time compared to software, for conventional, more deterministic domains.
3. Within the development team rather different disciplines are needed; for example marketing analysts, project managers, user (organizations), subject matter experts, educational technologists, system engineers, programmers, writers and editors, publishers, graphic

1. Two forms of semantic feedback are distinguished (Foley et al., 1990, p406). One form is presented to the user on completion of a requested function carried out by the application. The other form is presented to the user *during* interaction. The application sends a message to the user to let him know that a certain manipulation performed is semantically (not) correct, this can happen when least expected.

designers. Communication among the different disciplines can prove to be difficult as people have to learn to do this. Only a few senior developers have enough experience in such multi-disciplinary teams. The *leading role of the project manager* of such a team is difficult. He/she needs the characteristics of a director or producer of artistic projects such as a strong image of the product aimed at and the power to convince the team of their ideas. He should have a strong feeling for quality during all phases of the development process and should be able to evaluate many aspects of the design. She should be able to be responsible personally for the quality of all aspects of the design contributed by the different disciplines.
4. The organization of educational software development demands a different approach from that demanded from the organization of conventional IS development. To what extent are these demands different and how can they be implemented in a *project management style*?
5. The development of educational and instructional materials for all traditional and technological media is generally not based on economics but on idealism and the semi-commercial ambition of teachers, trainers and educational specialists. Only the production and distribution of the materials is planned commercially by publishers for the open market. Tailor-made instruction and educational software is mostly, as yet, not commercially successful for the user and provider. Measurement of success is difficult. This is also the case for expensive training systems (Rouse, 1992). Rouse presents several measurement issues, among others:
"Are the benefits of system use sufficiently greater than the cost?
Do organizations/individuals use the system?
Does the system solve the educational problem?
Does the system meet the requirements?
How do observers react to the system?"

In many cases cost/benefit analyses are strongly influenced by the accounting system used. E.g. quite frequently, the costs of (developing, implementing, maintaining) training personnel are not included in the training budget. (Rouse, 1992).

1.7 Research question and outline of the study

The problems with educational software development as reported in the literature form the most important reason for this study. The main objective of this study was to:

propose and validate criteria for better organization
of multimedia educational software development.

Communication between the disciplines and integration of the contribution of all skills involved into a qualified product are considered to be important subjects for study.

We compare the development process of educational software with the production processes of other "communication" products, e.g. film making. The reason for doing this is the following. In information systems design real integration of other disciplines into the team is not yet accepted. Acquisition of knowledge of domain experts is realized by sub projects or by interviews with the experts. User participation is becoming accepted in user interface and user manual design as part of systems design. In large projects it is as yet difficult to convince the information systems department that co-operation with the training department is necessary to improve application training of a new information system. With educational software, communication with the user is so important that communication disciplines may contribute a lot during the design process. Disciplines such as film making have a history of professional multi-disciplinary work. We focus on development methods for professional environments of different types, such as tailor-made educational software versus educational software for the open market and in-house development versus development with one or more third parties involved.

In chapter 1 the general research question is introduced and presented, see figure 1.6.

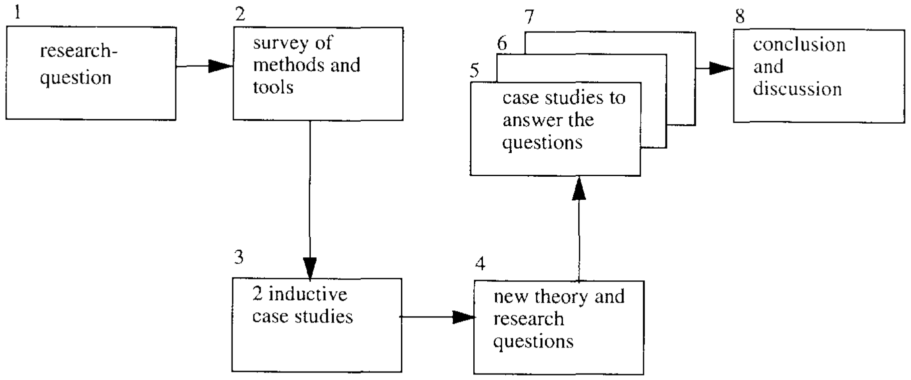

1
research-question
2
survey of methods and tools
3
2 inductive case studies
4
new theory and research questions
5
6
7
case studies to answer the questions
8
conclusion and discussion

Figure 1.6 Outline of the research project.

In chapter 2 a general framework is chosen for understanding and analyzing the development process of educational software. This framework is taken from the discipline of information systems development. Using this framework the literature on methods and tools for educational software development is surveyed. Although we focus on professional environments we survey several quite different methods and tools. Some are used in professional projects, others are only known in academia and in small projects in formal education; several different classes of tools are also surveyed using the framework.

In chapter 3 two inductive cases are analyzed thoroughly. The first case is a geography project for formal education. The second case is a project on an introductory multimedia course about the use of PC's. The first case concerns the production of highly interactive educational software. The second case concerns the production of a Teleac course consisting of TV programs, radio broadcasting, and software for exercises and exploration. The development processes of both cases are described and evaluated in terms of the framework for understanding IS development mentioned above.

In chapter 4 a new theory is presented and research questions are specified to evaluate the new theory of educational software development. The selection and the characteristics of the projects chosen for the case studies are accounted for.

Three case studies are described in chapter 5, 6 and 7 based on analysis of the project documents and on interviews with the project leader and some members of the development team. The cases concern training for bank counter employees; training for aircraft maintenance technicians; and training a second language for 8-11 year old children. At the end of each chapter the questions presented in chapter 4 are answered and the implications of these answers for the success of the product and the project are discussed.

The answers and the implications for each project are compared in chapter 8 and integrated into more general conclusions and guidelines for development of modern multimedia educational software products.

2. METHODS AND TOOLS

This chapter contains a survey of methods and tools for developing educational software. Before starting the survey a framework is chosen to describe the methods and tools in a consistent way. This framework is taken from information systems development. Its domain dependency is discussed before it is used to describe the domain of educational software. The methods and tools are representative of the state of the art in research and for practical use as far as is known from the literature and available products.

2.1 Introduction

Understanding educational software development requires a framework of concepts to enable assessment and comparison of methodologies and tools. Van Offenbeek (1993) investigated the usability of contingency frameworks for information systems development. "According to contingency theory an organisation should fit its environment and vice versa". The goals of contingency frameworks are to provide guidelines to tune the "approach of development" to the context of the project. It is not the method *per se* that is important but how a method is applied and implemented in the context (the result is called a "scenario" by Van Offenbeek). Van Offenbeek proposes a new framework based on three main variables: risk profile, strategy and outcome. Four contextual risks are distinguished: functional uncertainty (task), conflict potential (structure), technical uncertainty (technology) and resistance potential (people). The conclusion drawn in the study is that neither context nor approach influence success significantly but that "analyzing the tuning between context and approach results in a usable explanation" (p. 180). Contingency frameworks focus on the socio-organisational aspects of a project and on the way a method is applied in a specific context. This view may be important for practical situations, but we need to focus first on the basic intrinsics of methods for developing educational software.

Seligmann, Wijers and Sol, (1989) proposed a framework characterizing methods for information systems development, by a way of thinking, a way of modelling, a way of working, a way of controlling supplemented by a way of supporting (figure 2.1). The "ways of" are called the aspects of the framework. This framework can be seen as a descriptive way to summarize methods and tools. The "way of supporting" points to the tools and those aspects of a method that are supported. This framework has been used in many research projects on information systems development at Delft University of Technology (e.g. Wijers, 1991; Van Meel, 1994)

The framework is claimed to be useful for solving specific "methodical" problems and can be used as a first step when comparing "methods" or for performing an analysis between methods and tools. Wijers (1991) states (p. 13): "We emphasize the role of methods and automated tools in such processes[1] for information engineers" and "Several models are usually required for problem specification and solution in the application area, because specific features have specific significance at various levels of abstraction, and for different views of the problem area". Levels of abstraction and different views introduce various methods. With

1. i.e.: information systems development.

regard to the way of modelling e.g. Wijers describes four categories of modelling languages (p. 15-16). Wijers looks at the "various" ways without referring to the characteristics of any application domain, although the three cases he studied "concern the construction of what we call an information architecture." Wijers does not work out the characteristics of the domain and the type of systems under construction in the case studies. He only describes the dependency on the skills of the information engineers (p. 180) and the experts involved in the experiments which results in personalized strategies and a need for "individualized" tools (p. 181).

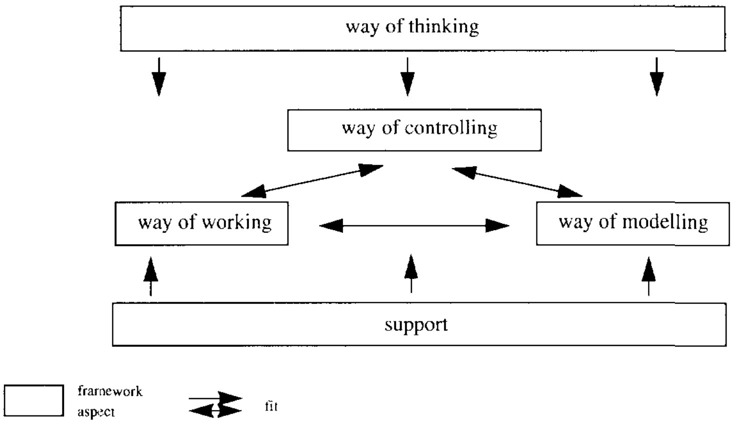


Figure 2.1 Framework for understanding information systems development according to Seligmann, Wijers and Sol (1989).

Of course there may be an invariant component, not influenced by any aspect of the domain, in information system development. There is some analogy in the design of an information system for financial administration (probably object-oriented when financial "objects" are the principal focus) and for training on the job skills (probably process-oriented when tasks to be carried out by the user are the principal focus).

We can, however, expect that the domain (and more specifically the type of application) is an important source of variance for the ways of thinking and modelling. Examples of different domains include: nuclear plant control, transaction processing, financial administration, legal expert systems, computer assisted design systems, desktop publishing, spreadsheets, videoclip design, and educational software. Characteristics of the domains and the software may differ greatly, Harel (1992) distinguishes two main types of applications: non-reactive and reactive systems. Non-reactive systems are data intensive whilst reactive systems show "behavior" influenced by an autonomous or independent real system. Some domains require embedded, strongly reactive systems and these systems are widely considered to be particularly problematic (Harel, p. 10). Harel discusses modelling reactive systems and analyzing the model. Educational software is a reactive system where interaction with the user is essential. One way to cope with the problem of developing such highly interactive systems is to separate interaction and domain semantics within an information system.

The Seeheim model (Green, 1985) is one approach for separating the domain and the user interface (UI) component during runtime. The Delft model (Van der Mast and Versendaal, 1992) describes such separation during development time to support the disciplines involved in application design with methods and tools suited for them (information analyst, software engineer, programmer, graphical designer, etc.). A serious shortcoming of the Seeheim model is, however, the poor support it gives for semantic feedback. Specifically, with graphical user interfaces using real life metaphors to represent domain tasks, semantic feedback[1] is needed frequently during the dialogue. The complexity[2] of an application (UI and domain component together) depends among other things on the intertwining of both components. We will present here a new taxonomy to order the complexity of applications. At least four classes can be distinguished with regard to this intertwining, see figure 2.2.

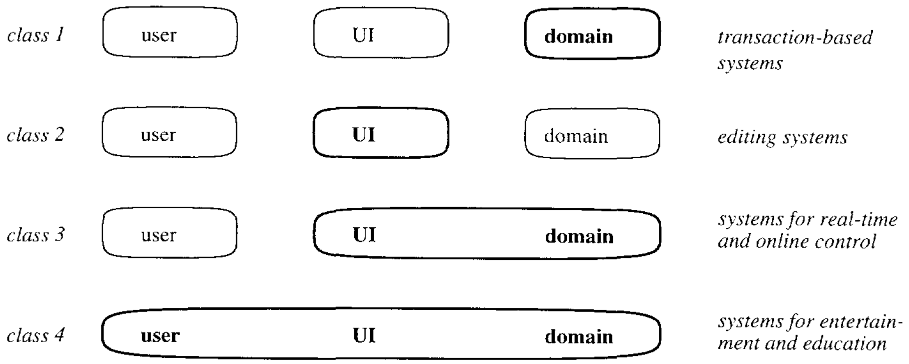

class 1
user
UI
domain
transaction-based systems
class 2
user
UI
domain
editing systems
class 3
user
UI
domain
systems for real-time and online control
class 4
user
UI
domain
systems for entertain-ment and education

Figure 2.2 Classes of applications based on the relationship user/user interface/domain.

One, non-reactive applications with functionally isolated domain and UI component, e.g. all conventual administrative and financial applications with simple transaction and command like user interfaces. The complexity is located in the domain component.

Two, applications where the domain mostly or only deals with the UI, e.g. graphical editors. The complexity is located in the UI component and the required datastructures.

The third class is that where there is poor isolation of UI and domain components and a very complex UI component where the user tasks are strongly structured and organized, e.g. process control systems with a graphical user interface. The complexity is located in the UI, the domain component and in the intertwining.

The fourth class is an application that has the domain positioned in the interaction itself where the user is, to a certain extent, considered to be part of the "domain", e.g. software for entertainment and educational software. The complexity is located and spread over all three components including the "user" and his behavior.

It is to be expected that the development of class four software introduces extra problems because the components cannot be distinguished at the semantic level of tasks the user can perform. Development can not be separated into easy phases for the three components (each with its own discipline). Development of class 4 systems requires a high degree of

1. Semantic feedback as part of user interfaces is feedback concerning the semantics of some aspect of the application itself, not of the dialogue. E.g some transaction is semantically not allowed by the application and this has to be presented to the user immediately during interaction.

2. In the view of the developer, not in the view of the user. Of course this complexity can also depend on the application semantics and the algorithms used. This is not the subject of this study.

communication and co-operation between several disciplines. Given this classification it is clear that class four applications need the most complicated architectures. Analyzing the intertwinement of the components and modelling different views is difficult.

The framework of Seligmann *et al.* is used as a device to classify the methods and tools used for educational software development in practical situations today and this framework requires analysis in greater detail.

Way of thinking
The way of thinking depends on the application area and its underlying perspective. Wijers (1991, p. 24) summarizes the way of thinking as follows:
"a way of thinking should define the assumptions of a method with respect to:
1. what constitutes an information system
2. what is the function of an information system with regard to its environment
3. what constitutes the environment of an information system
4. what are the major characteristics of the component parts of an information system and its environment."
For the purpose of this study on educational software, the skills and capabilities of the designers (5) and the attitudes of the users and consumers (6) may be added to this environment.

In the context of this study the way of thinking is also represented by assumptions as to whether social and organisational matters should be considered, as to whether information systems should be regarded primarily as computer systems or as systems of persons and computers.

These aspects of a way of thinking can be applied to educational software as follows:
ad 1. educational information systems can be considered to be a system consisting of users (students and teachers), the software and other media, e.g. textbooks, slides, videotapes and support services.
ad 2. to be a learning instrument for students alone or as an instrument for students and teachers together with different integrated modules, e.g. support for scheduling students and obtaining progress reports.
ad 3. the goals, objectives and contents of the surrounding curriculum can be included in the problem analysis and proposed solutions.
ad 4. how many classrooms, computers, time slots and teacher and supporting staff man hours are available?
ad 5. what are the skills and disciplines represented in the development team and how can these be improved?
ad 6. what are the ages, social background and attitudes of students and teachers? Is user participation during design planned?

Way of modelling
The way of modelling represents an interrelated set of modelling concepts, describing the types of models which can potentially be built (Wijers, 1991 p. 17). Modelling concepts must be chosen to specify the characteristic aspects of the application domain, the level of abstraction and the designers view.
Wijers (p. 15) discusses 4 general categories for modelling languages:
- free models (verbal and graphic) e.g. essays, storyboards, scripts, drawings;
- structured models (constraint by type of concept) e.g. structured scripts, diagrams, tables;

- mathematical models;
- dynamic models e.g. simulation models and prototypes for the analysis of dynamic behavior.

The interaction component of information systems can be distinguished, see 1.5, following Foley and Wallace(1974) and Foley at al. (1990). These authors prescribe conceptual, semantic, syntactic and lexical levels or models to specify the user interface of information systems. Interaction is crucial with educational software. So, a user interface-oriented way of modelling must be included. The levels of Foley are good candidates.

Way of working

Wijers (1991, p. 18) describes the way of working in terms of relevant tasks in the modelling process consisting of the components:
- possible tasks to be performed;
- decomposition of tasks into subtasks and decisions;
- possible order of tasks and decisions;
- informal guidelines and suggestions on how to perform tasks, and how to make decisions.

Of course development consists of more phases than modelling. In general, the way of working describes the process of developing.

Bots (1989) considers tasks as actions to be undertaken to achieve certain objectives. We assume that the base tasks needed to describe the way of working are the same for all application domains. Bots distinguishes the following relationships for the task structure in a modelling process (p. 39):
- parallelism between tasks;
- sequence of tasks;
- iterations of tasks;
- alternative tasks;
- optional tasks;
- synchronization of tasks.

Bots (1989, p. 44-45) describes two different views on task structure: as a hierarchy (describing the decomposition relationship) and a flow chart (describing the task-decision relationship). Using a concrete development process the task structure can be described accurately based on observations. Most development methods do not prescribe the structure of all tasks as obligatory; most methods allow some degree of freedom depending on the characteristics of the project and the resources. See, e.g. the overview given in Van Offenbeek (1993).

Way of controlling

The way of controlling concerns the management of the development process based on a way of modelling and a way of working. It includes planning and plan evaluation and regular project management aspects including quality assurance, Wijers (1991). Quality assurance (QA) is often a standard component of management methods and is considered to be related to a chosen way of working. Quality assurance is a written agreement to control many formally specified aspects that may be important or critical for the successful accomplishment of a development project. Both the attributes of the aspects and how to audit them are part of the agreement. Quality assurance implies that commitment is given to improve project conditions when necessary.

Way of supporting

Support for one or more of the other "ways" of this framework is possible. A few methods are based on the availability of a special tool. The belonging method is described in 2.2. Most tools, however, support the *programming* of educational software and are generally not dependent on the method of development itself. These tools are described and discussed in 2.3.

This framework of ways of thinking, modelling, working, controlling and supporting will be used as an instrument to analyse and evaluate the methods listed in the next section.

2.2 Survey of methods

To survey methods it is necessary to start with an overview of general opinions reported in the literature. Then a choice can be made of the methodologies to be studied in detail.

Developing educational software is an ill-structured problem where the solutions depend on user requirements. There is initially no definite criterion to test a proposed solution, much less a mechanizable process for applying the criterion. The problem space is not defined in any meaningful way (Simon, 1973). We do, however, know that the development process depends on the complexity of the desired solution. This holds for all cases, e.g. a teacher designing a small exercise for the pupil versus designing an educational software package as a strategic medium for all students studying a curriculum or for a company department.

Developing information systems is domain-dependent as described in chapter 1 (Giddings, 1984), however, developing products within one domain can also depend on the scale of the project, e.g. small versus large projects. At least one of the main conclusions to be drawn from many years of developing educational software may be that, in general, learning materials that have been programmed by teachers and trainers working on their own are of a much lower quality than those developed by multi-disciplinary teams (Rushby, 1992).

Without reference to any specific method, disappointing experiences in educational software development can be summarized as follows (Moonen and Van der Mast, 1987):
- in university environments the development process will be research-oriented and not product-oriented; products are often never used by anyone other than the developer;
- considerable educational software has been produced by individual teachers, lacking sufficient time, knowledge and expertise; often no formal commitment and man power is available for implementation of the product in the classroom;
- the team approach is an improvement over the individual approach but communication between team members from the different disciplines is difficult;
- teams of teachers failed to produce agreed scripts.

The most important common cause of these disappointments is the organizational aspect of the ways of working and controlling. Moonen and Van der Mast (1987) distinguish four classes of organizational approaches:
1. the individual approach;
2. the team approach;
3. the project approach, a team approach supplemented by strong project management;
4. the industrial approach, a project approach in which production of educational software becomes profitable for publishers operating in a market.

This classification claims to be applicable to the complete spectrum of environments in which educational software is developed and used, from schools to educational departments in business, and from small one person projects to multi-party projects where partners carry out different phases of a project from a commercial base.

Sherwood and Larkin (1989) present four other classes that focus on developing university courses. This classification is not compatible with the approaches of Moonen and Van der Mast

(1987). It is only usable for teams working on non-professional projects which are not covered in this study. It is interesting to see what kind of solutions are proposed to enhance the communication between different disciplines.
1. individual authors (teacher who designs and codes);
2. parallel teams (several individual authors); it is assumed that all team members are teachers experienced in their subject matter and at least in one of the skills needed for educational software (pedagogy, design and coding). Communication within the team is easy because they all "speak the same language".
3. serial team (teacher, designer, coder); different disciplines work together: the teacher starts with an idea, the designer molds this, and the coder implements it. There is a sequential production style.
4. mixed teams (teacher/individual author, designer, coder); in a mixed team the teacher of the serial team does some individual writing of the educational software, so he is better able to discuss design and coding issues.

Methods are selected for the survey which seem to support the team, project or industrial approaches. We will first describe some methods for development of educational software which are relevant to the spectrum of methods practised during the last two decades. After a short introduction each method is described in terms of the framework depicted in figure 2.1. The way of supporting belonging to a method is described only if a set of tools for support is part of the method.

In the next sections 8 methods are described which are representative of the state of the art. The list is far from complete but only a few special methods are well known for professional projects. Two methods have evolved from the use of standard software engineering methods, the Irvine-Geneva method from SADT (Kurtz and Bork, 1981) and the HOEP method from Yourdon techniques (Hartemink, 1987).

2.2.1 THE IRVINE-GENEVA METHOD

Bork (1981, 1985, 1987) proposed, during twenty years of experience of developing highly interactive materials for learning, a systems approach based on earlier successful methods for producing learning materials before the computer became available. His approach is based on organizing the work within a multi-disciplinary creative professional team. The team needs to focus on, among other things, the following issues:
- pedagogic design is most important, not information technology;
- excellent teachers are needed to provide design ideas and concepts, and to give feedback on preliminary designs;
- user interface design is considered extremely important;
- technical design must precede programming;
- apply general software engineering principles;
- use modern structured programming languages;
- avoid all authoring languages and systems;
- formative evaluation is an essential instrument for improving the design;
- funding of the projects should be at a professional level.

During the long period this method has been in use many aspects, such as pedagogic design, have shown little change (Bork et al., 1992). In this paper the following goals for development are presented:
- teaching material should be of the highest quality;

- teaching material should work for a wide variety of different students, aiming for mastery for all students. Learning should be interactive;
- teaching material must be concerned with motivational issues, so students will stay at difficult tasks;
- material must be produced within the times specified initially;
- materials must be produced within the costs specified initially;
- units should be adaptable to changing pedagogy and to changing technology;
- pedagogic design should not be influenced by implementation considerations.

Way of thinking
Bork assumes that educational software provides a "conversation" between a student and a teacher, where the student is at a computer display and the teacher conducts the dialogue through the medium of a computer program, Bork (1981). He states that a teacher designs the computer dialogues and that software development follows this. Bork also states that we need to consider new organizational patterns for the learning process. This would imply new course and curriculum structures, what he wants essentially, is fundamental innovation of the educational system in our societies using new technologies. He believes that the computer, although poorly used in education to this point, has great potential for making massive improvements in learning for almost everybody in the world. Bork seems to be rather negative about the current educational system (in the USA). His way of thinking is based on long term visions and on large scale professionally organized projects. Bork describes in many papers and books the need for excellent teachers, designers and other creative persons who are able to co-operate intensively in a peer team. His idealism is high, resulting in an approach that is usable in commercial artistic projects, i.e. with a combination of commitment, creativity and dedication. Bork strongly prefers professional skills and tools.

Way of modelling
Since the primary concern of this method is learning, the most important stage in development is that of pedagogic design which consists of two stages, overall and detail design. The design is recorded as a script, "a visual description of the programs behavior when a student runs it". Messages and pictures must be described on a frame by frame basis and full interaction and all possible branches are described using flow charts. The way of modelling given here is derived from the well known mature system for textbook publishing. Basically the script is a semi-formal but accurate visual programming language that consists of text blocks, sketches, flow diagrams and comments on all of these. Professional programmers will then write programs based on the script and competent artists will design the screen. In the same way as when illustrating a book, the artist will use his/her own non-standard method.

Way of working
After the subject matter has been chosen the author must prepare an outline (length e.g. 1 page) of the "mainline" approach in plain text. They can then continue on a frame by frame basis working sequentially through the program branches. A modified flowchart form may be used to describe the pedagogical design of a dialogue. This approach removes teachers from the computer details and enables them to interact with a variety of specialized helpers, secretaries, graphic designers and programmers.

The following tasks are carried out during development (Bork, 1984):
1. preliminary steps;
2. overall pedagogic design, review and revision of it;

3. detail design sessions, review and revision of the script;
4. screen design, review and revision of screen design;
5. design of code, review and revision of code design;
6. coding, review and revision of code; integration of code;
7. internal review and revision of the running program;
8. formative evaluation by target groups and expert teachers;
9. revision;
10. distribution and use;
11. summative evaluation after extensive use.

Way of controlling:
Many cycles of review and revision are built into the way of working. In the list of 11 points given above it is clear that review and revision is integrated in all steps and phases. In this way quality of the design is controlled continuously, not only at the end of the development just before delivery of the final educational product. The reviews are recommended to be carried out by external experts in the subject matter or the aspect to be reviewed (e.g. screen design). Reviewing must be supervised by the manager of the project (Bork, 1984).

Way of supporting
The Irvine-Geneva Course Development System is provided (Bork et al., 1992) to support management, pedagogic design, technical implementation, formative evaluation and improvement. Standard techniques and tools for project management are used in several of the design stages. During pedagogic design the script is entered, as soon as possible, into the computer using a script editor. The script editor supports version control and code generation. All tools work at workstations using Unix and X11. The way of modelling is supported basically by a directed graph[1] data structure, in which each node represents an elementary action. Once these directed graphs are entered and stored other tools can use these data for further refinement and code generation (synchronous editor, multilingual editor, version control). This support system is described in section 2.3.4.

2.2.2 THE TIME-SHARED INTERACTIVE COMPUTER CONTROLLED INFORMATION TV METHOD

The Time-shared Interactive Computer Controlled Information TV (TICCIT) system was developed in the early seventies as a system based on an instructional philosophy. The educational strategy of the philosophy is built into the system. It consists generally of a combination of drill and practice and tutorial learning styles (Merrill, 1980) to implement:
- the presentation of rules on the subject matter;
- examples of application of the rules and;
- means to practice them.

The student has maximal control over the sequence of the rules, examples and exercises to master the topics of the subject matter. The educational software and so the authoring has to be organized in an hierarchically ordered set of courses, units, lessons and segments. The segments contain the rules, examples and exercises. All segments are presented to students as maps to choose the next steps. The TICCIT authoring system was designed to be used by teams of educational experts. The concepts used in the TICCIT method were later refined into a general instructional design theory (Merrill, 1986).

1. A directed graph is a standard datastructure defined in computer science.

Way of thinking
Based on the learner controlled instruction paradigm (Merrill, 1980) the developer needs to analyze subject matter facts, concepts and rules and to construct adequate and wrong examples, practise exercises and test questions. This is based on a very straight forward way of thinking. Only one paradigm of learning is supported as mentioned above: presentation of rules, examples and practices. The TICCIT system assumes that, at the level of community colleges, learning means to learn concepts and rules.

Way of modelling
Specification of contents and educational strategies is based on the use of:
- lists of objectives, ideas and rules;
- storyboarding, which is the process of rewriting textual and pictorial displays so they will fit within the display limitations of the computer;
- flowcharts with all components, sequences and details;
- four hierarchical levels: courses, units, lessons and segments.

Way of working
The following 8 steps are recommended in the following order:
1. define the purpose;
2. collect resource materials;
3. generate ideas for a lesson;
4. organize your ideas for the lesson;
5. produce lesson displays on paper;
6. make flowcharts of the lesson;
7. program the lesson;
8. evaluate the quality and effectiveness of the lesson.

Way of controlling
No view of controlling the way of working and modelling is found in the literature, unless it is to carry out the steps in a prescribed order.

Way of supporting
The way of supporting is fully integrated into the way of modelling. Only one way of instructional modelling is supported: the TICCIT way. Several prompting structured editors are available to specify units, lessons and segments, and to fill in the contents of objectives, rules and examples.

2.2.3 THE INSTRUCTION MANAGEMENT PRESENTATION SYSTEM METHOD
The Instructional Management Presentation System (IMPS) of Godfrey and Sterling (1982) represents an attempt to simplify and clarify the long process of conceiving, designing, verifying, coding, testing and maintaining good educational software. IMPS can be used to create, the authors claim, "educational software on any topic, written in any computer language, to run on any machine". IMPS also provides a framework for educational software evaluation.

The main function of IMPS is to separate the programming and non-programming aspects of educational software. The process of developing educational software is divided into five[1] "layers" which are carried out in an iterative way. The IMPS methodology should work equally well in all organizational environments, e.g. schools, factories, etc. This method does not prescribe the realization of the designs but is limited to the design phase.

Way of thinking:
To know all about any given subject matter is difficult, if not impossible. Formalized education provides guidelines about what and when the learner needs to know. This may be based on theoretical ideas about learning, e.g. Skinner and Piaget. Developing educational software, however, emphasizes the imprecise nature of the "facts" of the subject manner. A certain amount of uncertainty is introduced (or revealed in) into education. Educational software is fundamentally a complex process. This means that developers need all the help they can get from other disciplines, especially information and software engineering.
The concept of separating programming and non-programming aspects of developing educational software is important.

Way of modelling
Concepts used to describe educational software include: curriculum, courses, goals, objectives, rules, examples and questions. Five basic teaching strategies are presented: drill, test, inquiry, simulation and tutorial. These are analyzed in terms of basic description, sequence, rules and examples, questions and feedback. This way of modelling is more descriptive than prescriptive. All teaching strategies are considered to be learning dialogues. Therefore most strategies (e.g. simulations) are not captured in detail by interaction. The method does not support more than the general design for these strategies.

At the level of local structures (see way of working) the use of flow diagrams is advised to analyse and define the underlying structures.

Way of working
The way of working is modelled in five layers which may be followed out of sequence. Former layers may be redone to improve the design. The layers are:
1. definition layer
Learning objectives and related rules must be defined.
2. local structure layer
Objectives and related rules can be ordered into structures. Examples and questions are added to the structures. A course may consist of a few rules and a few structures or of hundreds of rules and one structure. Structures are basic patterns of dialogue components.
3. presentation layer
To be dealt with as abstractly as with the two former layers. Devise prototypes of screen layouts on paper or using other media.
4. tracking layer
The analytic data for learner and teacher are described. E.g. objectives mastered, percentage of course completed, average response time, unanticipated answers.
5. mapping layer
The various ways that local structures and sets of objectives can be associated. Recommended and possible pathways. Adaptability of path for different learning histories. This is the

1. In Guildford and Sterling (1982) nine layers are described. The first five (definitions layer, local structures, presentation, tracking, mapping) can be considered as levels of modelling. The last four (student support, author support, site implementation, network implementation) present guidelines for the way of working.

pedagogic component depicted in figure 1.3.

Way of controlling
Not covered by this method.

Way of supporting
Not covered by this method.

2.2.4 THE GERY METHOD
Gery (1987) describes a method for the development of educational software for business applications that has three objectives:
- to assure timely production that achieves specified learning objectives;
- sponsor control of the product content, structure, design, screen displays and interactivity levels by continuous involvement and review, approval and sign-off on tangible paper based deliverables at the end of each major development phase;
- to control the development process (quality assurance).

Gery works out examples of all steps in all phases with special emphasis on the paper deliverables to be signed-off by the sponsor. From the examples it is clear that this methodology is only suitable for computer based training for simple concepts and not for computer assisted learning of higher concepts. Gery also gives guidelines for forming the development team.

Way of thinking
Gery defines educational software (called Computer Based Training CBT) within this business context as: "An interactive learning experience in which the computer provides the majority of the stimulus the learner must respond and the computer analyzes the response and provides feedback to the learner." The importance of interaction is emphasized strongly. CBT is, in this context, ment for well defined tasks e.g. in sales training, field engineering training and application software training.

Gery aims to be simple and effective with training skills, not teaching general concepts. As for the method itself it is assumed that developing educational software has many parallels in conventional software development. There are, however, differences in complexity, non-linearity, and the individual nature of the learning process.

Way of modelling
Gery presents a matrix of levels of interactivity and learner control. Three classes of CBT are described: case study presentation, interactive presentation and application software simulations. It should be possible to find an appropriate solution for a training problem using the three classes.

Way of working
Gery describes 8 phases and many steps for each phase.
1. project definition;
2. high-level design;
3. detailed design;
4. scripting or storyboarding;
5. programming;
6. testing and evaluating;

7. production and distribution;
8. administration.

Way of controlling
A list of CBT development roles is given: project manager, program sponsor, instructional designer, subject matter expert, media expert, graphic designer, learner evaluator, production administrator, CBT administrator. Requirements for skills and activities are listed for all roles. Guidelines for project management, quality assurance (QA) and financial control are provided.

Way of supporting
No supporting tools other than a wordprocessor are mentioned. For programming the best language or system must be chosen according to criteria on a decision list.

2.2.5 THE SHIVA METHOD
SHIVA[1] is a system based on a specific method (Elsom-Cook, 1992). The method is described by describing the concepts of the system. SHIVA was developed with support from the European Union. SHIVA is an authoring system based on the integration of two components: ECAL, an intelligent authoring tool which provides adaptive instructions and ORGUE, a graphic educational software development tool. The educational software is designed on the basis of a concept map consisting of high level objectives in the domain and the structure of the course. At this high level SHIVA contains the components typical for intelligent CAI systems: student model, knowledge presentation and interaction model. The student model represents the student as a subset of the concept map. Each concept has numeric attributes which are changed (positively and negatively) during interaction. A knowledge base can be specified by the author using visual editors representing networks and attributes.

These concepts (objectives) are linked to sets of multimedia "frames" of material. At this lower level a visual flowchart editor allows the specification of branched sequences of frames similar to traditional frame based educational software.

According to the first evaluation of SHIVA the results were poor. It is difficult to evaluate effectiveness because it has not as yet been used on a large scale in realistic development projects for educational software.

Way of thinking
SHIVA is a multimedia authoring system that facilitates the work of multi-disciplinary authoring teams both on the high level of teaching objectives and at the low level of media editors.

Way of modelling
Knowledge of the subject matter domain is modelled as concepts with only four abstract attributes: connectedness, relatedness, importance and generality. This knowledge is represented as a network on the screen. The concepts of this knowledge are associated with "frames" of learning materials. Two types of frames exist: presentation and diagnostic frames. The contents of a frame are described and specified in flowcharts. The student model is represented as a subset of the concept network, where each concept has a numeric attribute (value between 0 and 1) indicating the ability and speed of mastery of the learner. The dialog

1. The acronyms SHIVA, ECAL and ORGUE are not explained in the publications studied. The SHIVA project is project 1010 of the Delta program initiated and funded by the European Union.

model is based on the idea of "focus" in conversation.

Way of working
The author team is recommended to produce a set of educational objectives in terms of subject matter, target group, prerequisite knowledge and pedagogic strategy. This specification has to be accomplished to the level of each individual frame and this must be done off-line, on paper. Editors can then be used to input the subject matter and the control structure separately.

Way of controlling
All materials are stored in one central authoring environment with many editors and debugging tools. No explicit advice is given for control.

Way of supporting
The SHIVA method is represented implicitly in the SHIVA system. The following supporting tools are necessary to use the method. It has no sense to use the method without using the system, cf. the TICCIT method and system.
- a flowchart editor, the intended internal sequence of material components within a "frame" can be specified;
- a concept map editor, teaching objectives can be specified from lists of keywords;
- a link editor, frames and concepts can be associated;
- debugging and simulation tools, learner behavior can be predicted;
- a number of editors are available to create text, video, sound and animations, the smallest components of a frame.

The SHIVA system was not available for evaluation.

2.2.6 THE ENVIRONMENT FOR DEVELOPING AND USING COURSEWARE METHOD

Environment for Developing and Using Courseware (EDUC) is an experimental method with supporting tools for developing tutorial software. Basically, it is an attempt to relate methodological principles (ways of thinking) for design and development to ways of working and supporting (De Diana, 1988). In particular, De Diana has integrated, to some extent, design methods for traditional instructional materials and software engineering methods. His hypothesis is that *developing educational software is more than developing instruction material plus developing software.*

As a result of analyzing 40 references from the literature on developing traditional instructional materials he reports the following characteristic core tasks for development:
- definition of outcomes;
- content analysis;
- specification of learner attributes;
- definition of instruction strategies;
- development of materials;
- try out and revision.

The software engineering life cycle is recognized as:
- requirement analysis and definition;
- system and software design;
- implementation and unit testing;
- systems testing;
- operations and maintenance.

The following are given as general requirements for EDUC:
- support for the complete life cycle;
- smooth transfer between phases;
- the method should be easy to repeat;
- the method should be clear to teach and to learn;
- the method and its tools should improve the productivity of the development team.

EDUC is used by educational technology students for several courses on educational software design and development.

Way of thinking
The following entities are recognized for development of educational software:
- student;
- content matter;
- response analysis;
- representation of learning results;
- decision algorithms for the student's path.

Several aspects of educational software are recognized. Static aspects cover the structure of and the relations between content topics. Dynamic aspects describe the way the instruction process is organized, distinguished into instruction tactics and instruction strategy. Tactics concerns the way the contents are presented and strategy covers aspects of the dynamics (the dialogue). Instruction tactics can be compared with conceptual and lexical aspects according to Foley (section 1.5), while strategy can be compared to syntactic aspects. De Diana (p. 86) uses the method of Moran (Moran, 1981) for specifying conceptual, communication and the physical components of the design. These components show some analogy with the conceptual, semantic, syntactic and lexical aspects of Foley et al. (1990).

Way of modelling
One general template for tutorial educational software is used for lessons to be developed using EDUC. The template consists of blocks for contents, student, answer analysis, scoring and decision. In EDUC structures and the content of the educational software are designed separately. A tutorial educational software product is structured hierarchically using 4 levels: network and node (both abstract definitions), component (dialogue specification) and content (specification of text and graphics to be presented to the student).

Way of working
The method of Moran (1981) is used to organize the development process. The following two phases are important during the design process:
- the conceptual component consisting of
    task level; specification of lists of learning tasks for the student;
    semantic level; specification of lists of entities and concepts to be manipulated for learning;
- communication component with
    syntactic level; specification of the dialogue in a formal way;
    interaction level; specification of the lexical form of all objects to be manipulated during the dialogue.

Development is done as a smart combination of top-down refinement and bottom-up evaluation and integration. The design has to be mapped into a structure of networks, nodes and pointers, components and contents. Several editors are available to specify the mapping of a design into the EDUC system.

Way of controlling
EDUC does not provide an explicit way of controlling. De Diana writes that the specification of a lesson can be checked for wrong pointers and wrong internal syntax of the specification commands. This is not a facility for control at the semantic level, but control at a technical level.

Way of supporting
A matrix of simple tools is provided for the phases specification, construction, formative evaluation and prototyping, on-line guidance is provided.

2.2.7 THE HOEP METHOD

The Handleiding Ontwikkeling Educatieve Programmatuur[1] (HOEP) method is considered to be the ancestor of a small family of methods that are used for many projects in the Netherlands. It is a comprehensive method. Therefore, and because the author was involved, at project manager level, in the initial definition (Van der Mast, 1989) the HOEP method is described in detail. The PRINT method (section 2.2.8) and POCO method (chapter 7) are descendants and improvements of the HOEP method. The HOEP method is used in the "Sunshine" case, discussed in chapter 3.

In the context of the Dutch National Informatics Stimulation Program (INSP) a method was developed and applied in some pilot projects to develop educational software in a professional way leading to an industrial approach (Van der Mast, 1986 and Van der Mast et al. 1991; Moonen and Schoenmaker, 1992). This method is an adjustment of several professional, well known, methods for information system design, software engineering and project management (DeMarco, 1978; Jackson, 1979; Yourdon, 1979). Special attention has been given to the separation of functional design and technical realization. The interface between both is a well documented script that includes some prototypes.

This method was developed by experienced educational software designers and has been used in several projects. It is independent of hardware and software environments. The method is published in the Handbook edited by Hartemink (1988). Applications developed using the method have been published in a series of reports of strategic projects in the INSP. A coursebook as a Guide for Self-Study (Van den Camp et al., 1988) has been published with one case worked out on paper and as an available prototype. The HOEP method has been applied explicitly to a few large model projects of the INSP. As the method and experience of its use is well documented, it has been used to provide a framework for enhanced methods in later national educational software programs and in some projects for computer based training in the commercial sector.

Way of thinking
The need for project management principles is considered a most important base for this method. Project management is the vehicle used to control the contribution of and the communication between different disciplines: subject matter expertise, educational technology, media design, user interface design, software engineering, programming and last but not least scheduling of budgets and time. The framework of the method is based on conventional methods for information system design, project management and software

1. In English Manual for Developing Educational Software.

engineering. It is assumed that educational design is a special case of functional design in system development. It is also assumed that it is not (yet) possible to involve designers who are experienced in both educational design and technical design. Therefore integration of both disciplines should take place by different teams communicating via documentation, prototypes and a minimum of personal contact. The act of conceiving educational software, that is, sketching its distinctive behavior and appearance apart from the mechanisms of its implementation is recognized as a distinct professional activity. In this method, however, less comprehensive guidelines are given for the educational design than are given for the technical design.

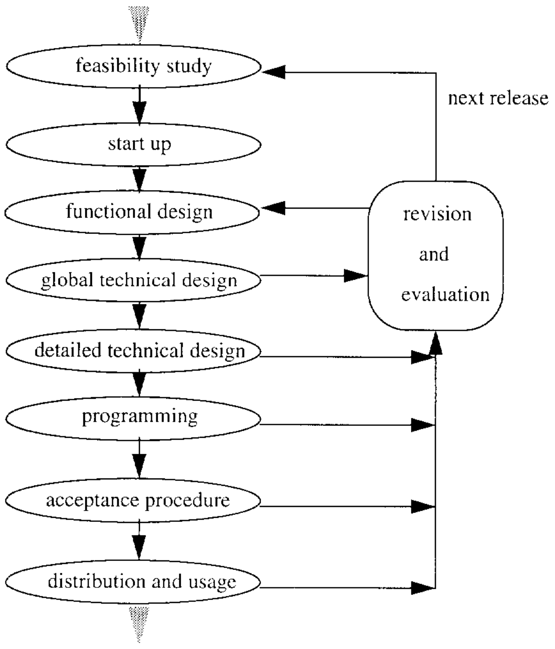


Figure 2.3 Overview of the phases for educational software development used in the Handbook for the Development of Educational Software (HOEP).

Way of modelling
The educational design is based on a task-oriented approach. Each learning task should be described using composition diagrams (see section 3.1.3). The functional design translates the contents of the educational design into program functions. The sequence of these functions is specified in the user interface design. The Yourdon method is prescribed with use of dataflow diagrams, state transition diagrams, procedure hierarchy diagrams, entity relationship diagrams and a data dictionary (Yourdon, 1979; DeMarco, 1979). Moonen and Schoenmaker (1992) say about the HOEP and similar methods, "Perhaps the emphasis... has been focussed too exclusively toward the software engineering aspects.".

Way of working:
Overall project management is divided into five components: organisation, phasing, planning, quality assurance and documentation. Each component has the same set of aspects that must be

specified: a goal, a final product, activities, disciplines and skills needed within the team.

The following main phases are provided, see figure 2.3: feasibility study, start up, functional design, global technical design, detailed technical design, programming, acceptance procedure, distribution and usage.

During these phases the questions why, what, when and how are analyzed and answered. The feasibility study results among other things in a curriculum analysis and an outline of the solution. Functional design, resulting in the script, has to be done in a strongly iterative way and must be based on an in-depth analysis of all relevant available information. Decomposition of the functional design is given in table 2.1.

1. describe the requirements
2. specify the complete educational design
3. specify the system functions
4. specify the user interface
5. specify the acceptance test
6. design textbooks and all other media

Table 2.1: Decomposition of functional design

Way of controlling:
In HOEP guidelines are given for controlling the development process with a QA-plan. This includes writing the QA-plan itself, designing reviews and audits, procedures for problem solving during the project and procedures for changes in the design. Examples are given in section 3.1.

Way of supporting:
As HOEP has been published as a method for large scale use within the INSP program and beyond it provides guidelines on how and when to use tools to support some phases of the development process. It describes (including examples) how to use spreadsheets for project control, a script editor for educational design and prototyping, other prototyping tools, case tools for technical design and a special library with educational procedures (section 2.3.3). Use of the general programming language C is advised.

2.2.8 THE PRINT METHOD

One of the follow ups of the Dutch INSP (Van der Mast et al., 1991) was the Project Invoering Nieuwe Technologieën (PRINT[1] project) (1989-1992). The Ondersteunings-structuur Courseware Ontwikkeling (OCW[2]) was founded to provide and implement quality control for all educational software projects (at primary and secondary school level) and to support some educational software development teams with less experience in project management. The members of OCW gained their experience during INSP and they started to apply the HOEP method (see section 2.2.7), where possible, to the project proposals submitted by employees from schools and educational institutes. The highest priority was given to the development of

1. Project Invoering Nieuwe Technologieën (PRINT) is Dutch for "Project Implementation New Technologies". The author was not involved in PRINT.
2. Translation into English "Courseware Development Support Service".

courses and workshops for the teams to be formed for the projects. Courses were given on the following subjects: project management, educational design, techniques for educational design and specification. Further courses were developed and given on the technical subjects: introduction MS Windows, designing on-line help, prototyping, Visual Basic and user interface design.

A series of documents with guidelines for the projects were published and used during the courses (PRINT 1990; 1992a; 1992b; 1993). All project proposals submitted to the PRINT management were required to use these guidelines. Based on the HOEP method the PRINT method separated the design by subject matter experts resulting in a "script" and realization by software houses. This approach proved to work well for the majority of the PRINT projects (PRINT, 1993, p. 49). This method differs in two aspects from the HOEP method: one the "feasibility phase" of the HOEP method is changed or omitted and two, the "functional design phase" of the HOEP method is slightly changed and renamed the "educational design phase" to distinguish educational design work from technical design work.

Way of thinking

The way of thinking is the same as for the HOEP method. The main principle is the emphasis put on project management. The rather linear process of phasing and reporting is also the same. It is assumed that project proposals containing good ideas come from within schools and educational institutes. The persons who proposed a project were supported, when necessary, and the progress of the project was controlled by the OCW. An OCW staff member was responsible for the quality assurance of each project.

Way of modelling

A way of modelling is prescribed for the educational design phase. First a data dictionary must be made that consists of a description of all relevant concepts and definitions from the educational approach. Dataflow diagrams must be used to describe what should happen at the higher levels starting with a context diagram of the system as a whole. The first refinement from the context diagram is called the Leerstof Oriëntatie Schema[1] (LOS). As yet media selection has not been made. Computer Leerstof Oriëntatie Schema[2] (COLOS) should be derived from the LOS-schemes to specify the media selection results. When these models have been specified a Lestaak Analyse Tabel[3] (LAT) must be composed and a Courseware Actoren Tabel[4] (CAT). Instructions and examples of how to create and fill-in the tables are given in the published guide (PRINT, 1992b). An alternative for the tables can be used if the structure of the educational software is not sequential (or at the level of detail where the individual freedom of the student comes up). This alternative is the composition diagram as prescribed in the HOEP method. During the specification phase dataflow diagrams are again used in addition to state transition diagrams to specify the dialogues. The realization phase is mostly performed by a software house using its own techniques and software engineering methods.

Way of working

The first five phases are performed under the control of PRINT. The result is a complete working product consisting of educational software and all associated materials, however, not

1. Translation: Subject Matter Orientation Scheme.
2. Translation: Computer Subject Matter Orientation Scheme.
3. Translation: Lesson Task Analysis Table.
4. Translation: Courseware Actor Table.

as yet in a form suitable for commercial distribution. The sixth phase, distribution and maintenance is, according to the aims of the PRINT project, left for the educational publishers operating in the Dutch education market. The phases are derived from the HOEP method as follows (see table 2.2):

PRINT method	HOEP method
1. project proposal phase (GO/NOGO)	feasibility study (GO/NOGO)
2. project plan phase (GO/NOGO)	start-up phase
3. educational design phase	functional design phase
4. specification phase	functional design phase (continued) high level technical design phase
5. realization phase	detailed technical design phase programming phase acceptance phase
6. distribution and maintenance phase	distribution and usage

Table 2.2: Relationship between the phases of PRINT and HOEP method

1. Project proposal phase
This phase is where the initiative is taken and a proposal is submitted to PRINT management. The educational and organisational requirements and some tentative solutions are described. A budget plan is part of the proposal.
2. Project plan phase
In this phase the feasibility of the accepted project is investigated and possible solutions are worked out, enough details are given to allow a decision to be made by experienced PRINT staff. The main lines of the quality assurance plan are already worked out and it is possible to cancel the project during this phase.
3. Educational design phase
This phase gives a clear educational picture of the result aimed at and *how to use it in practise.* During this phase, in most projects, a representative of a software house and an educational publisher are added to the team to improve communication in the later phases. In most cases the product of this phase is delivered to a software house for specification and realization. Sometimes the software house receives the result of the specification phase depending on the characteristics of the product under development and the available expertise at the designing institute. Sometimes a prototype is included with the documentation of this phase.
4. Specification phase
In this phase the contents and the details of the software are described. The phases functional design and technical design from most software engineering methods are included in this phase. The user interface design is made during this phase.
5. Realization phase
Detailed technical design and the programming and testing is done during this phase followed by necessary improvements or revisions. PRINT uses a QA-plan to judge and to accept a product using educational and technical criteria.
6. Distribution and maintenance phase
This phase is carried out by educational publishers and is outside the control of the PRINT

project. The publisher adds his logo and other standard messages and prints the associated materials (text and exercise books) in the house-style. A marketing campaign is organized.

Way of controlling
The emphasis is put on quality assurance from the project plan phase onwards. In PRINT (1990) guidelines are given for organizing the quality control process through all the phases. Guidelines and standards for all required documents are given and the guidelines, based on those described in the HOEP method, are very complete. Of course the quality of the implementation of these guidelines depends on the expertise and commitment of the PRINT management team reviewing and controlling the progress of a project.

Way of supporting
No special tools are prescribed by the method. Only the method and the guidelines are given to the project teams. All notation techniques can be carried out using a pencil and paper. Sometimes special editors are used for the educational design. There is a free choice of prototyping tools. The realization of the design is up to the software houses, however, all software has to run on MS Windows PC's, with "standard" specifications as available in Dutch schools.

2.2.9 CONCLUSIONS
Examination of the various methods yields some general conclusions. One should, however, realize that any method cannot be considered in isolation from the organisation (i.e. the people) implementing it. It is also important to remark that some methods have never been used in anything other than small or academic experiments, while others have been used and evaluated in large scale projects. A classification of the methods is given in table 2.3.

	thinking	modelling (educational)	modelling (system)	working	controlling
Irvine-Geneva	teacher-based	y	n	(y)	y
Ticcit	theory-based	y	n	n	n
IMPS	instruction-based	y	n	n	n
Gery	manager-based	y	n	y	y
SHIVA	technology-based	y	y	n	n
EDUC	style-based	y	n	n	n
HOEP	manager-based	(y)	y	y	y
PRINT	manager-based	y	n	y	y

Table 2.3: Classification of some methods within the framework of Seligmann et al. (1989), y: present; n: not present; (y): hardly present.

The first general conclusion is that some methods (Bork, IMPS) give more emphases to personal quality and optimal co-operation within the design team (i.e. the implicit way of working) than techniques and formalisms for ways of modelling and supporting. Other methods (TICCIT, SHIVA, EDUC) place the emphasis on ways of modelling and supporting. *Only two methods are explicitly dedicated to one instructional model of learning (TICCIT and* EDUC). No method supports the classification and design of the user interface explicitly. Last but not least, only one method tries to bridge the gap between instructional and software design explicitly, but it did not succeed in achieving this goal[1] (EDUC). The general conclusion to be drawn is that the attention paid to the different "ways of" is not well balanced. Either the way of modelling (TICCIT, IMPS, EDUC, SHIVA) or the way of working (Irvine-Geneva, Gery, HOEP, PRINT) gets most attention; thinking, and controlling are not well highlighted. No method fulfills all "ways" of the framework completely. HOEP and PRINT seem to be the most complete methods listed.

All the methods that have been used for large, and serious projects have had only partial success. For some methods, success with the technical realization phases has been reported (Van der Mast, Hartemink and Henkens, 1991). The same paper also reports problems with the educational design phase and with communication within the teams and with low skill levels of the designers.

Aspect of framework of Seligmann et al. (1989)	Attention found in survey of methods
way of thinking	is neglected in most methods except Irvine-Geneva
way of modelling	is highlighted by most methods but in different ways
way of working	is highlighted by most methods in different degrees
way of controlling	is neglected by most methods except the HOEP family and Irvine-Geneva
way of supporting	is highlighted on an ad hoc base except by the methods that are based on authoring tools

Table 2.4: Overview of the results of the survey of methods.

Moonen (1990) concludes that neither professional and industrial team nor specific individual approaches have been able to keep development projects within time and according to specifications agreed upon beforehand. Developing educational software seems harder than "hitting a moving target". The main problem seems to be *communication*. On one hand there is the problem of communication among team members from different disciplines and on the other hand there appears to be a problem in communication between the developers and the target group, as embodied in the time-consuming formative evaluation aspect. According to Moonen (1990) educational software is "too new" and teachers are unable to describe and to agree upon what is exactly needed beforehand. The required amount of review and evaluation sessions is too high and therefore too expensive for this kind of product within the regular educational system. Moonen pursued two options to resolve this dilemma of formative evaluation: to give more emphasis to formative evaluation or to find ways to make formative evaluation less important. The first option resembles the approach of developing products for the consumer market. The second option resembles the "approach' of free artists working for themselves and hoping to get enthusiasm from other people or the "approach" of a tailor-made

1. EDUC is an experimental system for research purposes. It has been used in a controlled experiment by educational technology students at Twente University.

production process where use is almost guaranteed.

Finally we can conclude that neither method highlights *all* aspects of the framework well. The results of this analysis is summarized in table 2.4.

2.3 Survey of ways of supporting

Tools to support some aspects or phases of development are rather independent of the development method used. They will be analyzed and compared in section 2.3.8 with respect to the support they provide for the framework of Seligmann et al., see figure 2.1. This survey will not cover the practical use of the tools in the realistic context of projects but only the supporting power of the concepts embodied in the tools. Some tools are described only briefly because they are used by many developers and contain straightforward features for programming and prototyping. Some other, research-based tools are described more extensively to explain their more comprehensive concepts, these are however, as yet not used commercially.

In support of information systems development we can distinguished non-automated and automated tools. Non-automated tools are such things as paper and pencil and other traditional "off-line" media. Automated tools are obtaining increasing importance. Criterion for effective support is a way of supporting that fits neatly into the ways of modelling, working and controlling.

Wijers (1991) gives the following list of minimum types of support. CASE[1] tools should offer:

- management support	(way of controlling)
- process guidance	(way of working)
- model construction	(way of modelling)
- model verification	(way of modelling)
- model derivation	(way of modelling)

Liker et al. (1992) introduced the concept of "user support infrastructure" for computer aided design (CAD), consisting of:

1. formal and informal user training;
2. quality of user support (formal, informal and documentation);
3. environment for ongoing learning;
4. user input to CAD decisions.

They also emphasize that the real power of support tools comes from "informing", i.e. using the computer to add new information to the process of creating and decision making. They emphasize the role that (national) culture plays in the way information technology is introduced and managed. Therefore some "factors" may operate differently for different types of users and work contexts.

From Liker at al. it can be concluded that not only the "functional"[2] aspect of supporting is important (e.g. a drawing tool or a verification tool) but that the "intellectual"[3] aspect is even more important for effective support of human creative work.

The development process of a communication product (as is educational software) depends strongly on quality control to guarantee that the users (students) of the final product get the right learning experience. As with other communication products (e.g. film) tools are needed

1. Computer Aided Software Engineering
2. Functional means the analytic engineering done mostly by young, relatively inexperienced, engineers.
3. Intellectual means the creative engineering done mostly by senior experts.

"...to make it always possible to get a good feeling of the current best guess of what the audience's experience would be" (Heckel, 1991, p. 208). As in UI design, educational software design can be supported by the use of scenarios. In educational software a design scenario is normally called a script (Ibrahim et al., 1990). Nielsen (1990) presented a taxonomy of scenario's for user interface design that consists of a classification using the criteria purpose, medium and source of inspiration. The taxonomy is depicted in table 2.5.

	Text	Storyboard	Running system	Text	Storyboard	Running system
Communi-cation	Illustrative scenarios	Extended illustrative scenarios		Documenta-tion (man-ual)	Presenta-tion scenar-ios. Documenta-tion (video)	Documenta-tion (tuto-rial)
Structure thinking	Refined design sce-narios			Design sce-narios		
Testing	Archetypi-cal interac-tions	Iterative test of mock-ups (paper)	Iterative test of mock-ups (running)	Experimen-tal setting, generic test suites	Mock-ups (paper)	Mock-ups (computer)

Table 2.5: Scenario types classified into the three dimensions: source of inspiration, medium, and purpose (Nielsen, 1990). Left columns inspired by empirical observations and right columns inspired by designer's ideas.

Nielsen evaluated paper and computer mock-ups of a single user interface for a videotex system to find usability problems when using mock-ups. His conclusion was that both types of mock-up can be used for heuristic evaluation but that there are differences in what usability problems are easy to find in the two cases (Nielsen, 1990, p. 320).

Development can be divided into design and realization. Tools to support development are in most cases focussed on the realization phase, especially when programming prototypes or final products. Some authors mention that good tools are necessary to produce good educational software (O'Malley, 1991). Some tools are ment for individual authors (e.g. cT, Sherwood and Larkin, 1989) while others are particularly designed for a team with different disciplines (e.g. Sodoyer, 1990). Some tools are ment for one specific phase (narrow tool) whilst others can be used for subsequent phases at the same time, supporting consistency of design over phases.

In literature, however, the importance of tools for the creation of educational software is frequently overestimated, while educational and pedagogical requirements and skills are often underestimated: the quality of educational software is "created" during the early conceptual phase. This is supported by the fact that, though modern hardware and software offers "good" facilities and tools, "many early development problems continue to plague current efforts". (McDermott, 1990). Liker et al. (1992) draw the same conclusion concerning the use of CAD tools in a comparative study of the use of CAD systems in the USA and Japan. The way the tools are being used by the designers (in Japan more as a medium between senior and junior designers) is significant *not* the quality of the tools (in the USA more professional and sophisticated).

Authoring tools are languages and systems that are used by authors to design and realize educational software (McDermott, 1990). In our opinion, however, authoring languages and authoring systems do not support the creative process of *design* because they have many technical and conceptual constraints that general programming languages do not have. So, all the features/limitations of such systems do not contribute to good designs. The "simplicity" of authoring often belies the complexity of effective instructional pedagogy (McDermott, p. 200). Bork et al. (1992) argue that "the use of authoring languages and authoring systems is an unfortunate direction".

The traditional features of authoring environments are among others (McDermott, 1990):

- user assigned logic and sequence	(modelling);
- page/frame orientation	(modelling);
- template formats	(modelling);
- menu orientation	(modelling);
- embedded management	(modelling);
- open ended authoring	(modelling);
- code generation	(working);
- base-language bridges	(working);
- assisting authors pedagogically	(working);
- transportability	(working);
- integration of peripherals	(working);
- storage	(working);
- execution speed	(working).

From this list it is clear that authoring systems support only the well-structured (Bots, 1989) modelling phases and the way of working with programming. The modelling phases during the initial conceptual creative process are not supported at all by these systems.

Some tools and systems are presented in the next sections. At the end of each section an evaluation of the tool is given.

2.3.1 MODULAR CAI SYSTEM DELFT

The Modular CAI System Delft (MCSD) was developed in the seventies as a research platform to support all seven main functions (see section 1.2) involved in the development and use of educational software (Wiechers, 1975; Van der Mast, 1981). The separate modules of MCSD provide support for one or more functions each. All modules perform their functions as independently as possible from the others, only exchanging data encoded in ASCII. Portability and flexibility have the highest priority.

The most important aspect of MSCD is the form in which the educational software is distributed for presentation to the student. This form has the nature of a language in which all necessary actions (e.g. display a text, display a picture, move cursor, set clock, analyze student response, test a condition), resulting, in total, in about 50 basic instructions (Van der Mast, 1982) are represented by a set of elementary instructions. These instructions and their operands are interpreted during run time. The language is called CAI ASsembler COde (CAIASCO). Like other assembler languages humans find it difficult to read. CAIASCO is an intermediate code generated by development tools and by a compiler. Interpreters exist for different operating systems, mainly for MS DOS and Unix. The lessons developed in the seventies for minicomputers can, technically, be followed today on a graphic workstation without any change.

Modules are available for all main functions, see figure 2.4. In the beginning MCSD consisted of two components. One for developing and one for presenting educational software. The PLANIT author language and the PLANIT authoring system for mainframes were chosen as support tools for the design and the production phase. The PLANIT system (Bennik and Frye, 1970) provided an interactive tool for producing educational software of the frame-oriented type without programming. Using a prompting system the several types of frames could be filled in and the paths could be defined by many commands or with special decision frames. The lessons could be followed in student mode. A listing of lessons in PLANIT code could be printed for documentation. It was also possible to write lessons in PLANIT with a standard editor and to load the file into the PLANIT system. The PLANIT system itself was implemented in Fortran and could run only on the large mainframe computers of the time.

It was decided, as a first step for a portable presentation system, to define the CAIASCO code and to build a PLANIT compiler. The first version was implemented on the same mainframe computer as the PLANIT system was running on. The other component of MCSD was the lesson interpreter.

The first CAIASCO interpreter was built for a single user operating system DOS/BATCH for PDP11. Later versions were implemented for other minicomputers, MS DOS, Macintosh and finally Unix. Today MCSD is only run on Unix and MS DOS computers.

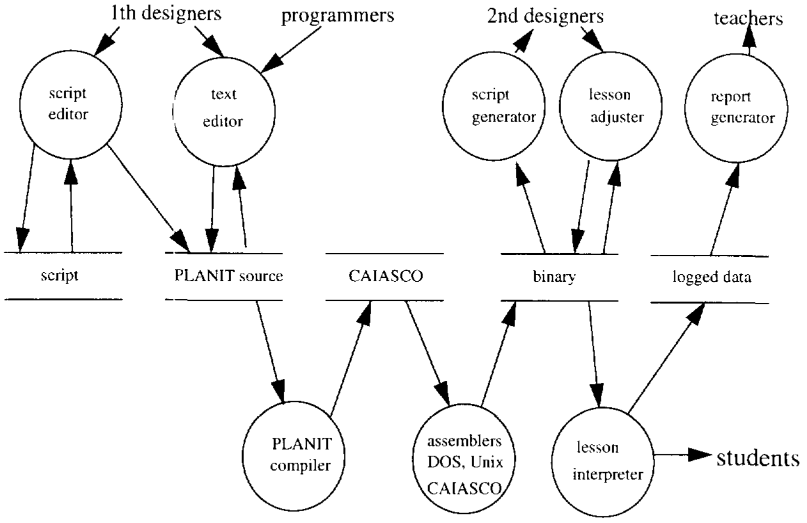


Figure 2.4 Dataflow diagram of all computer supported functions within MCSD. The "1th" designers are the initial designers of the educational software. The "2nd" designers may be teachers or other persons who wish to change the contents of the runtime versions of the educational software slightly.

MCSD had the feature of storing restart points during sessions to continue later without redoing any part of the former session from the beginning. This was also a feature of the PLANIT system and language. An on-line calculator could be controlled by the lesson (constants, variables and functions could be loaded and functions of the calculator could be prohibited for use by the student). During sessions the path followed and all (un)anticipated responses were recorded for evaluation purposes. Messages could also be sent to the teacher during the student sessions. Later, a tool was developed to select data from these records and to generate reports about sessions for formative and summative evaluation of the lessons.

Given the important role the teacher plays in applying educational software and the demand to exchange existing lessons with other teachers it is necessary to provide the teacher with a means to adjust or modify the contents of the lessons slightly (Van der Mast, 1988). The Lesson Adjuster was developed for this purpose, it provides a mode to follow the lesson by interpretation of the lesson code on PC. It is also possible to switch to an inspecting and an editing mode to see and look at all details of the lesson and the anticipated responses. It is not possible to change any part of the structure of the lesson. What is possible is to edit all text, colors and layout on the screen and also the strings representing the anticipated responses to adjust answers and to specify new synonymies and extra anticipated answers. The changed lesson can be disseminated directly to students in this format. In practice the Lesson Adjuster proved to be a tool not only for refinement of but also for maintenance of lessons.

In the early 80's it was felt that all existing supporting tools for development of educational software with MCSD were only supporting the programming and not the designing. A new conceptual model was then proposed: the script editor.

The script editor (Sodoyer, 1986; Sodoyer and Van der Mast, 1987; Helwig, 1987) is a tool that supports modelling during specification of educational software. As a script is the result of the specification it must mention among other things: the educational objectives, instructional strategies, structures within the subject matter, layout definitions, dialogue specifications (including the anticipated answers of the student), interaction prototypes and instructions for the programmers. All these descriptions need to be very detailed.

The script editor supports the creation and printing of scripts and visualization of the contents for on screen prototyping and review. A script is a structure of different components that are connected and filled in during the design phase. Some types of components are forms with predefined fields that can be filled in a free format as documents. Other types of components represent screens with adjustable layouts to specify the subject matter and anticipated reactions, feedback and branching instructions. The user can add own forms and screen layout formats to the library of standard components. During review in the visualization mode, comments can be added (and inspected) for all components and fields. Using this feature communication within a design team can be improved. Writing a script does not require any programming. Some attempts were made to realize a feature to generate PLANIT code from the script but this failed. The reason for the failure was that real scripts contained too many textual instructions in natural language. Thus, the script editor is only viable for design support.

In practical situations a printed version of the script is delivered to teachers along with the binary code for the lesson. The teacher can then use the script to understand the structure and the contents of the lesson. This has helped some teachers to adjust educational software using the Lesson Adjuster. As scripts made during the design phase contain a lot of comments and instructions for the developing team a decision was made to develop a script generator (SG) to generate a paper script directly from the binary version of a lesson having the same structure and formatting as the scripts made by the script editor. Of course this kind of script does not contain any comments made during the design phase. This script is a practical paper document for teachers, it can be used to prepare lesson changes using the Lesson Adjuster. After changing a lesson a new version of this documenting script is generated by the teacher.

In 1984 a PLANIT compiler and lesson interpreter were built on the Unix platform, compatible with the MS DOS versions; MCSD was therefore superseded for developing new educational software. This was because of the modest features of the author language PLANIT and the advantages of new programming environments such as Borland Turbopascal for MS DOS. It was thus decided to re-implement the script editor (SE)[1] for Unix and the window systems X11. In the beginning scripts made using the MS Dos version of the script editor could be loaded into SE to be edited further with more features for graphics, A/V components and animations. Later SE was changed into a powerful independent tool to investigate the support of the pedagogical design of educational software (Soerjadi, 1990). SE is described in section 2.3.2.

Evaluation
MCSD is a toolbox consisting of slightly coupled modules. The modular approach to development and the run-time system gives rise to a high degree of portability over different hardware and software platforms and this was achieved with a modest level of technical functionality. The choice of an author language was not successful. The Script Editor for designers and the Lesson Adjuster and the Script Generator for teachers are the most interesting modules. Experience with the Script Editor showed that it is difficult to support designers with electronic scripts. MCSD also showed the difficulty of continuing support over design phases. MCSD supports informal modelling and to some extent supports, implicitly, the way of working, but, as a tool it does not contribute to achieving a higher quality of the final product.

2.3.2 SCRIPT EDITOR

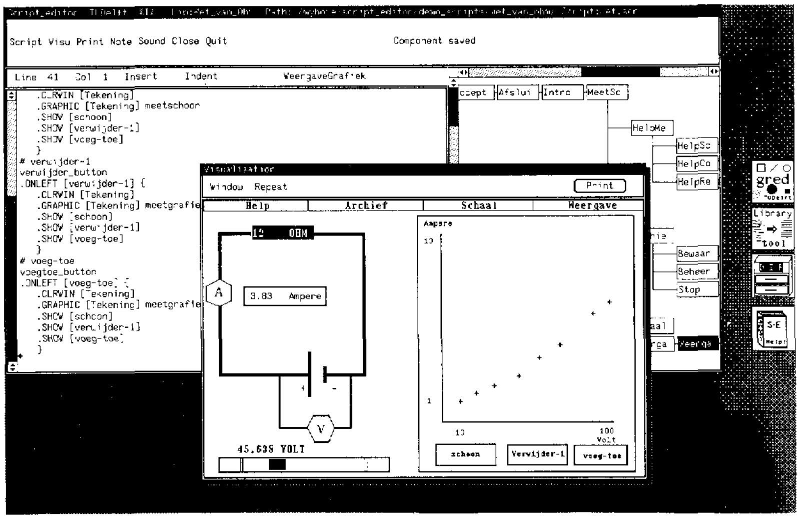
This began as a module of MCSD (section 2.3.1) but Script Editor (SE) can now be considered to be an independent tool that is used to support the early design and prototyping phases of the development process (Sodoyer, 1990). SE was designed to support the ways of modelling and working by a team of developers. It is a generic tool used to fill in scripts as spread sheets are filled in. The basic cells and structures must be specified first. The structures define the attributes and relations between screen areas. The designer can then compose and fill lesson components and order them in pedagogically chosen sequences. This is done without a programming language. Visible objects are specified by direct manipulation on the screen. Relations between the objects are specified using a command language, see figure 2.5.

The basic cells can be structured documents for communication between designers or template screens for students. Interaction on the student screens can be specified using a command language working on instances of the basic cells. Dialogues and multimedia animations can be created. So, running prototypes are part of the script and can be stored and distributed at once. Specified interactions can be presented in a window besides the script for prototyping. Using a note editor colleagues within the team can attach comments to any object on the screen of the running prototype. All main concepts: documents, structures, lesson maps, graphics, dialogues and animations can be inspected and edited on the same screen in parallel, see figure 2.5. All data are stored in one file system so that version control of all related data can be organized easily. SE produces complete scripts that can be used by programmers for implementing the design with minimal communication to the designers (Soerjadi, 1990).

Evaluation
SE supports the pedagogical design phase by providing integrated tools for structured documents, hierarchical structures of components, specification of standard layouts, rapid prototyping of dialogues, management and control of standard dialogue components stored in libraries, and reviewing and annotating designs by many people. It is based on the principle that design and programming should be separated completely. Thus, SE supports the way of modelling a lesson in frames and also the way of working and controlling during the design

1. The acronym SE is used for the Unix version of the script editor exclusively.

phase. It is intended to be used by professional teams in which all the necessary disciplines are represented. The electronic scripts produced have proved to be sufficient guides to technical design by programmers using author languages (cT) and general programming languages (Turbopascal).



```
    .CLRWIN [Tekening]
    .GRAPHIC [Tekening] meetschoon
    .SHOW [schoon]
    .SHOW [verwijder-1]
    .SHOW [voeg-toe]
    }
# verwijder-1
verwijder_button
.ONLEFT [verwijder-1] {
    .CLRWIN [Tekening]
    .GRAPHIC [Tekening] meetgrafie
    .SHOW [schoon]
    .SHOW [verwijder-1]
    .SHOW [voeg-toe]
    }
# voeg-toe
voegtoe_button
.ONLEFT [voeg-toe] {
    .CLRWIN [Tekening]
    .GRAPHIC [Tekening] meetgrafie
    .SHOW [schoon]
    .SHOW [verwijder-1]
    .SHOW [voeg-toe]
    }
```

Figure 2.5 Screen dump of a session using the script editor SE. All features can be used simultaneously: editor for specifying standard components and storing these in a library, structure editor for components, text editor for documents, layout editor for screens, graphics editor for screens.

2.3.3 EDUCALIB

Educalib is a library of functions, procedures and supporting programs for the realization of educational software on MS DOS computers. The aims of Educalib are to save time, to standardize and to provide technical portability. Educalib is written in Microsoft C. It was developed in 1987/1988 for the Dutch INSP to develop educational software to run on NIVO-PC's[1] these are used in Dutch secondary schools on a large scale.

Educalib offers features for:

- window management;
- menu management;
- writing to the screen using automatic formatting;
- reading from the keyboard;
- text editing in adjustable screen areas;
- presenting graphics;
- analysis of dialogues using open questions;
- database management;
- on-line scaling and printing;

1. The standard PC of the NIVO project within the INSP was at the time an XT with 640 kB and a cga-monitor. NIVO is a large sub project of the INSP. NIVO stands for Nieuwe Informatietechnologie in het Voortgezet Onderwijs, in English: New Information Technology in Secondary Education.

- double linked list datastructure management;
- error management.

This library was used for many large projects within INSP (e.g. in the project FIETSPLAN (Van der Mast et al., 1991a) and the *Sunshine* project, see section 3.1) and in some POCO-program[1] projects (POCO, 1992).

It was decided at the beginning of the POCO program (see chapter 7) to implement educational software for MS Windows. To give support to other platforms the commercial product XVT was chosen. XVT is a software layer between the operating system and application for programming the user interface. The MS DOS version of XVT only supports text output. A graphic MS DOS version of XVT was developed by POCO, that also supported the POCO standard graphic user interface in the same way that Educalib does. Products made with this library are portable from MS DOS to Windows offering the same user interface. Within the POCO program 24 products were realized using the XVT library. These products can be ported to Windows, Macintosh, Windows NT and OS/2 (POCO, 1992).

Evaluation
Educalib version 2.0 was made in the period before Microsoft introduced the mature Windows 3.0. The features offered by Educalib were accurately tuned to the requirements of a number of relevant educational software projects of the time. The features for open dialogue programming were very slanted to the requirements of educational software. The performance of the library was good for the processors of the period. As an experiment, an existing course written in PLANIT for MCSD was converted to C with Educalib, see Geuchies and Sleeuwenhoek (1989). They concluded that programming in C with Educalib was easy for professional programmers and that the procedures provided in Educalib were adequate and used frequently. The XVT library was a better solution to the same problem. Both Educalib and XVT only support the way of working during realization.

2.3.4 IRVINE-GENEVA CAD/CAI SYSTEM

This system has evolved over 25 years and is driven by pedagogical needs (Ibrahim et al., 1990). The pedagogical design strategies are similar to those of the early years, see section 2.2.1. Ibrahim et al. state "Since a good CAL lesson is mainly a dialog between the learner and the computer, the major building blocks of the "scenario" deal with program output, user input and how the program should respond to user input. All other operations are indicated in free text as instructions to the coder". The main goal of the system is to facilitate the task of educational designers who specify the lessons and the task of programmers who implement, test and maintain the source code associated with the specifications[2]. The main tool of this system is a script editor working at a workstation under Unix and X11. A script is a semi-formal visual specification for modules. The tool supports: script entry, script update, version control, partial automatic program generation. The data structure created by the script editor is a

1. For more information on the POCO program, see chapter 7. The *Woordwijs* project studied is part of the POCO program.
2. The system is called in later, unpublished articles IDEAL (Interactive Development Environment for Active Learning. It is a joint product of the Educational Technology Centre of the University of California at Irvine and the University of Geneva. The system is used at both sites. At Irvine only the script editor is used without code generation. The implementation was carried out by a private company. SE (section 2.3.2) was developed with the same goal in mind. With SE code generation was not a success. The electronic scripts including real prototypes of details and comments for the programmers are made available for the programmers on-line. Thus, separating design and programming.

directed graph in which each node represents an elementary action. The representation of a script is basically a kind of special flow diagram with several different kinds of nodes. The automatic code generator produces ADA source code. The generated code has to be completed by hand. Directions to the programmer can also be incorporated within the script.

Evaluation

This tool was intended to support the educational design with script editing and technical realization with partial code generation, however, from technical reports it seems that only the *specification* of educational and user interface design are supported. The system also supports communication between designers and programmers by providing the facility to add instructions for the programmer to the script. The prototyping facilities are not impressive and in any case not "rapid". It is doubtful of this kind of tool helps to improve the quality of the designs and the efficiency of the development process.

2.3.5 AUTHORWARE PROFESSIONAL

Authorware Professional is a multimedia authoring tool with a graphic user interface, figure 2.6. The educational software is designed by creating and connecting icons representing components of text, graphics, video, sound, animation, etc., for presentation to the students and components for measuring a student's performance and controlling the flow of the course. It is especially suited to tutorials, drills and non-model-based simulations[1]. The computing features are basic. The features that support digital video are excellent. The system runs on Macintosh and Windows platforms with good compatibility of the course structure and not too specialized presentation features.

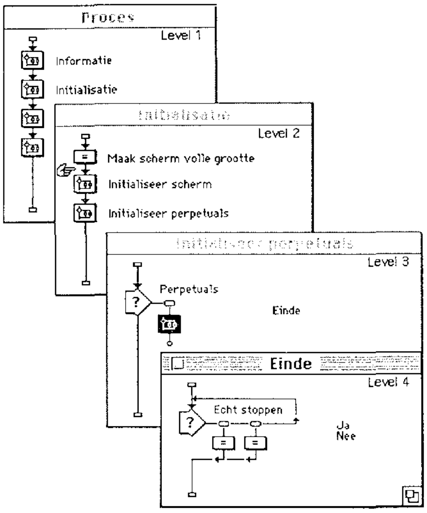


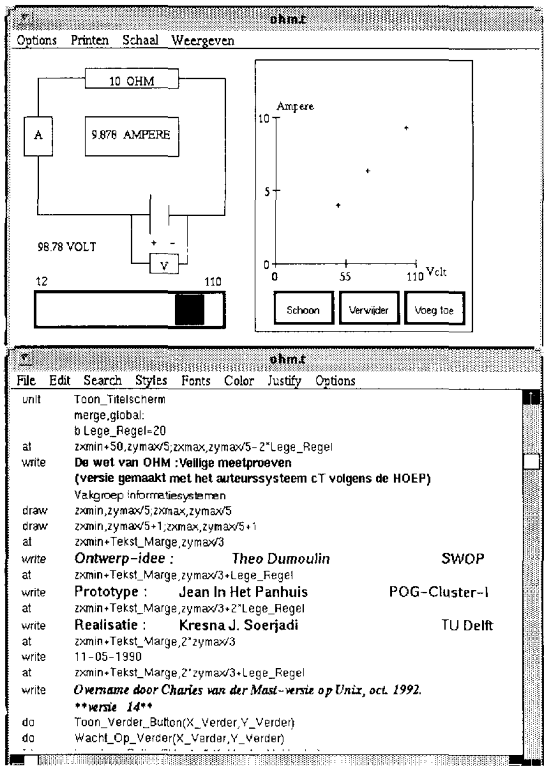
Figure 2.6 . Example of the graphical representation of components and structures and the decomposition in levels possible with Authorware Professional. (Buve, 1992).

1. The system has no computing facilities. Simulations based on mathematical models cannot be realized.

Evaluation
This system is rather new, and the experience of a large number of developers is not yet available. Non-programmers find it easy to develop new modules and prototypes. Programmers can realize more components later and link them to the prototype step by step. As Authorware Professional is strong in working with AV components it is important to organize the media management of up to some hundreds of content files carefully. The system supports rapid prototyping and realization of the final product. So, it supports mainly the way of working during realization, but the feature to specify components at a high graphic level offers a slight opportunity to support educational modelling. This tool was used in the case presented in chapter 6.

2.3.6 CT

The authoring system and language cT, see figure 2.7, is based on a special philosophy for developing educational software. Sherwood and Larkin (1989) present the possible organization styles in the following spectrum: individual author, parallel team, mixed team and serial team and investigate the disadvantages of the team approach for development, see 2.1.



```
unit    Toon_Titelscherm
        merge,global:
        b Lege_Regel=20
at      zxmin+50,zymax/5;zxmax,zymax/5-2*Lege_Regel
write   De wet van OHM :Veilige meetproeven
        (versie gemaakt met het auteurssysteem cT volgens de HOEP)
        Vakgroep informatiesystemen
draw    zxmin,zymax/5;zxmax,zymax/5
draw    zxmin,zymax/5+1;zxmax,zymax/5+1
at      zxmin+Tekst_Marge,zymax/3
write   Ontwerp-idee :        Theo Dumoulin              SWOP
at      zxmin+Tekst_Marge,zymax/3+Lege_Regel
write   Prototype :     Jean In Het Panhuis          POG-Cluster-I
at      zxmin+Tekst_Marge,zymax/3+2*Lege_Regel
write   Realisatie :    Kresna J. Soerjadi               TU Delft
at      zxmin+Tekst_Marge,2*zymax/3
write   11-05-1990
at      zxmin+Tekst_Marge,2*zymax/3+Lege_Regel
write   Overname door Charles van der Mast-versie op Unix, oct. 1992.
        **versie  14**
do      Toon_Verder_Button(X_Verder,Y_Verder)
do      Wacht_Op_Verder(X_Verder,Y_Verder)
```

Figure 2.7 . Screendump of the source program of a lesson (below) written in the author language of cT and the running lesson (above).

The range of quality in materials produced by a team is quite narrow (not too bad and not very good). The opposite is the case for individual authors. Communication within a team is both a strength and weakness. They conclude from extensive studies that in the steady state of experienced individuals and teams, teams take more than twice the total number of person-hours to produce one unit as do individual authors. Sherwood and Larkin appear to be one of the few promoters of the individual approach. Their striking example is the production of a textbook: it seems to be normal that the publishing company has to typeset a book, however, desktop publishing tools have become sufficiently user-friendly today that a growing number of authors find it advantageous to typeset their own books. Sherwood and Larkin studied many tools for individual authors of educational software and made a comparison with cT.

According to Sherwood and Larkin the ultimate ideal is an individual author who knows about all aspects of development. They compare developing educational software with developing textbooks: thousands of professors write up their lecture notes, one hundred sets are reviewed by publishers, ten are actually published and three books sell enough to return a profit for the author.

The author language cT is suited to faculty use at universities for creating sophisticated educational programs (Sherwood and Andersen, 1993). This language system allows overlap of design, implementation and testing. The system is available for Macintosh, Windows and Unix machines (Anderson and Sherwood, 1991). The system is language based, see figure 2.7. The cT programming language is a granddaughter of Tutor, the authoring language developed within the context of the PLATO project in 1967. The authoring system provides an integrated environment for programming in the cT language and editing graphics, animations and multimedia without the need for programming.

Evaluation

cT only supports the production phase. It offers a programming language with many features very suitable for educational software. The programming environment enhances the possibility of rapid prototyping and producing a final product without interruption. cT gives portability of lessons over the platforms Macintosh, MS DOS, MS Windows and Unix with X11 as an important technical extra, (Soerjadi, 1990).

2.3.7 TENCORE

TenCORE is a set of authoring products for DOS, MS Windows and OS/2. The set consists of Producer, a complete multimedia authoring system for novice and experienced users, CMI, a mastery-based testing and record keeping system and LAS, a Language Authoring System for the advanced developer of educational software.

Producer (version 2.1) is an authoring system with a graphic user interface for direct manipulation. It is possible to lay out displays with graphics, text and multimedia directly, to combine them with questions and menus, and immediately have a working lesson. The objects and templates which can be created are based on the features of the author language LAS. Integrated editors are available for character fonts, graphics, animations, audio, bit mapped images, data and student records. Lessons made using Producer can be integrated as modules in the CMI and LAS members of the family.

CMI[1] has a menu-driven user interface to create and deliver tests, to manage assignment of multimedia learning resources, and to monitor and report on student performance. Test questions created with Producer and LAS can be used. Student records can be exported to standard database systems. Reports can be generated on individual and group performance.

1. CMI version 4.2.

Templates are provided for creating open answers, multiple choice and matching test items. Test questions may be chosen randomly from a pool of items.

Producer LAS[1] is the authoring language of the system. The language is derived from the Tutor Language of the Plato system and has the same structure of units, commands and tags. It is a block structured programming language with hundreds of variables, commands and tags based on common instructional terminology (e.g. <answer>, <compare>, <wrong>, <okspell>). In practise some users use Producer for making the main design of a lesson using an educational expert and ask programmers to fill in the more complex parts and units using LAS. The command editor is the main editor of LAS. Commands can be chosen from pull down menus and parameters of tags can be typed in directly or can be generated from direct manipulations on screen (creation of graphics, cut/paste text, etc.). A graphics editor, a character font editor and a bit-mapped image editor are available among other things. Commands for video and audio can be integrated with all other commands.

Lessons are stored in source format. During authoring each unit can be run by a built-in interpreter. For better performance the lessons must be converted to a binary format for presentation to the student. TenCORE is not available for Macintosh and Unix workstations.

Evaluation
The TenCORE system is a professional support tool for a wide range of users. It provides a good set of tools for almost all the tasks of a development team, however, it supports the educational design phase only with its (not rapid) prototyping features. No educational strategies or structures are embedded. This tool is ment for the realization phase of a project. It is an open flexible system, as a general programming language, with features for improving the productivity of the developers. The facilities for product management and maintenance seem to be modest for large strategic educational software projects where software engineering and quality control are needed. The source code of large lessons is difficult to read if you do not apply software engineering principles. Due to the very large number of variables, commands and tags in the language it takes a considerable amount of time to become an expert on this system. Standard programmers also need to acquire this expertise.

2.3.8 CONCLUSIONS

MCSD, SE and Irvine-Geneva are tools (as far as is known at present) that have no commercial background. These tools were not developed on a commercial base. The tools Educalib, Authorware, cT and Tencore are commercial products, all but Educalib are used on an international scale. This means that these tools provide features that satisfy, at least to some degree, the demands of the developers market. All tools differ on many points.

A summary of the characteristics of the tools described here is given in table 2.6 where they are scored for the way they support the ways of modelling (by subject matter expert, educational technologist and programmer), the way of working and the way of controlling.

Educalib is a specialized programming library with special procedures for educational software. Such libraries support only the software engineers and programmers of the team. If the library does not support the features required for a product they have to be programmed in a general programming language. New features can be built into procedures to be added to the library for a series of related educational software products. Such a library does not support the creative designers of the subject matter, the didactical strategy and the attractiveness of

1. Producer LAS version 5.0.

interaction. Moreover, it impedes design if programmers try to convince designers to "design for the features" offered by the library.

System	Modelling SME	Modelling educational	Modelling technical	Working	Controlling	Communi-cation
MCSD	script	script	no	no	standard Unix facilities	no
SE	script	library of didactic components	no	separation design and realization	standard Unix facilities	designers/ reviewers/ program-mers
Educalib	no	no	educational procedures	conventional software engineering	no	no
Irvine-Geneva	script	script	partial code generation	conventional software engineering	standard Unix facilities	designers/ reviewers/ program-mers
Authorware Professional	picture based animations	flowcharts	technical prototyping	not	no	no
cT	mathematical models and simulations	authoring language with many functions	student mode for rapid prototyping	all work by one person: no	no	no
Tencore	no	authoring language with many functions	technical prototyping	no	no	no

Table 2.6: Comparison of tools. Features for modelling are scored for use by the subject matter expert (SME), the educational technologist (didactic) and the programmer (technical).

The tools Authorware, cT and Tencore (and many others available on the market) support programming, each on a different way. cT and Tencore provide programming languages with special statements for educational use and media presentations which can increase the productivity of the programmers. These programming languages have constraints for the use of data structures and other elements important for software engineering. An important advantage of these tools are their prototyping capabilities. Prototyping supports modelling by the designers. The Authorware system does not have a real "programming language" but a graphic specification style for decomposition and ordering of the functional components.

The features provided for support of the communication between developers of different disciplines are mentioned. Only SE and Irvine-Geneva explicitly support teamwork with a note editor to comment scripts. SE also provides prototypes embedded within the script in aid of other designers, external reviewers and programmers. Authorware Professional may facilitate the communication between disciplines implicitly because of the graphic specification style.

It is striking that almost all commercially available tools[1] provide good prototyping

1. Authorware Professional and cT were analysed for this study, but many similar tools are available commercially.

features. The prototypes created can be evolved to produce the final educational software product. Some of the non-commercial tools provide support for the early modelling phase by subject matter experts and educational technologists, but they do not really support the realization of the final product. One can say that these tools are based on separation of design and realization. No tool supports the early design phases better than traditional media (paper, transparency, pencil, video) and isolated rapid prototyping software tools do, nor the tool *per se* provides support, but *how* it is used within the team. The skill and the craftsmanship of the team is best supported by tools that support internal, between individual designers and different disciplines within the team, and external, e.g. with users and students of the target group, communication on design ideas and details. A tool should support passing the baton on within the relay team, Liker et al., (1992).

The MCSD system shows that old attempts to support all aspects of the development process are too complex and that this approach is not successful in practical situations.

From this survey it can be concluded that tools are not a critical resource for successful development of educational software. Depending on the method used and the skills available the tools may contribute to success, if expectations are not too high. This conclusion is not based on an analysis of practical use of the tools mentioned but on the fact that the very existence of many different, isolated tools supports the general assumption that tools can only support smart developers in the same way as paper and pencil or overhead projectors support smart designers. Therefore, the availability and use of specific tools is not considered to be a critical factor in developing educational software.

3. ANALYSIS OF TWO PROJECTS

It can be seen from the survey in chapter 2 that most of the methods and tools for educational software support only a limited number of aspects of the framework of Seligmann et al., 2.1, and only one or more isolated steps of the development process. They all seem to have a narrow monodisciplinary "bandwidth". Two projects were analyzed and evaluated carefully by studying documents and by taking interviews to improve insight into the developing process. The projects were finished before this investigation took place. The author participated in the first project as the managing "problem owner". The framework of chapter 2 was used for the case studies.

The first project (*Sunshine*) was selected because it was a "pilot" project of the Dutch national program INSP. Extra resources were available for choosing and applying a development method explicitly, for providing optimal conditions and skills for the development teams, for internal and external summative reviews of the final product and for external audits on the development process. The subject of this project is formal geography education at secondary school level.

The second project was a television program about using your PC at home (*PC-Privé*). In this project software, radio and TV programs, a textbook and a telephone service for the students were produced. These had to be strongly integrated with respect to contents and time scheduling. This project was interesting as a non-conventional case that was oriented strongly to the consumer market. It is not an educational software project in *sensu stricto*, however, the same media and disciplines are integrated to produce one market-oriented product with less emphasis on interaction than found in educational software.

The aim of the analyses of these projects was to investigate whether the aspects of the framework could be recognized or not and how this may have influenced the development processes.

3.1 The SUNSHINE project of the Dutch Information Technology Program

3.1.1 INTRODUCTION

The Dutch Information Technology Stimulation Program INSP was carried out from 1984-1989 (Van der Mast, 1991) and the author took part as a project manager of Cluster 1, see below. INSP, in total, included the business, research, education and public sector. The "education" section of INSP was allocated about 15% of the total budget. In this thesis when we refer to the INSP we mean the "education" section. This education section was allocated 250 million guilders for a period of 4 years. The main goal was to stimulate the introduction of information technology at all levels of formal education. INSP was structured as a set of projects organized in five clusters.

Cluster 1 focussed on the construction of a national and regional infrastructure for the development and distribution of educational software.

Cluster 2 carried out projects for various school sectors, distinct policy priorities were established in each sector. The following sectors were supported by separate budgets: primary education, lower general secondary education, higher general secondary education, special education, formal and informal vocational training and adult education.

Cluster 3 covered the measures taken for in-service training of teachers and for provision of information to schools.

Cluster 4 covered the measures for pre-service training of teachers.

Cluster 5 dealt with educational research with regard to the introduction of information technology into schools.

Content-free and infrastructural work was concentrated in Cluster 1. The priority goals for Cluster 1 were:
- to design and implement an infrastructure for the professional development of educational software, paying particular attention to standardization;
- to establish three centres for educational software design;
- to establish an infrastructure for the registration, testing and evaluation, dissemination and distribution of educational software;
- to initiate expensive projects with general generic contents, such as CD-Interactive and expert systems.

These four goals were worked on using a number of projects listed below:
- to construct an infrastructure for regional development centers;
- to provide a design method and tools for developing educational software;
- to train educational software developers;
- to provide professional design environments and standards for target computers in schools;
- to support some pilot projects with professional development teams;
- to evaluate and disseminate educational software available on the market;
- to provide educational programming languages and environments for schools;
- to experiment with videotex and networks for distribution and communication;
- to experiment with interactive video disks.

The role of the author was to select or initiate, approve and supervise all projects according to the general goals and within the given budget, in co-operation with a counterpart at the Ministry of Education and Sciences (Van der Mast, 1989; Van der Mast, Hartemink and Henkens, 1991).

Strategy

By 1984, there was a considerable body of experience, both in the Netherlands and in other countries, with the development of educational software. Many projects, however, resulted in unfinished products, products unusable outside the very small circle and context of the developers, products that other teachers did not like, and products that were almost impossible to maintain although revision was a constant necessity.

Due to these difficulties it was concluded that the difficulty of developing and implementing educational software in schools had been badly underestimated. This was also made clear by the first project proposals for the INSP budgets. Goals were described and detailed incompletely, phasing and planning were poor while planning and budgets for programming were omitted in most cases. All proposals seemed to be based on the "do-it-your-self"-approach for users, because of the many disappointing experiences in the past the project management of Cluster 1 decided that the projects needed to be organized and equipped in a professional manner. Projects that had important long term perspectives were granted. A strategy was chosen in which a learning process was started by a set of related projects (*pilot*-projects) with an emphasis on quality control and external audits. A secondary aim was to investigate whether a professional development style for educational software for the Dutch school system was possible. The Sunshine project was one of these pilot projects.

Other aims of this strategy were:
- to carry out investments for the post-INSP period;
- to integrate the needs of all participants: teachers, students, designers, programmers, projectmanagers, publishers, schools;
- to give a high priority to training these participants;
- to use modern and professional methods and tools;
- to give a high priority to publishing and disseminating the results;
- to try to interest commercial software companies.

The first assumption for the development method was that it should be based on the professional project management principles used by software producers at the time.

3.1.2 ORGANIZATION

As the Sunshine project was one of the pilot projects extra funds were provided to assist the project in achieving much more than just a package of educational software in itself. The main goals of the project were (Van den Camp, 1988):
1. to try out the HOEP method (see section 2.2.7);
2. to apply some available design and implementation tools;
3. to evaluate the method and tools;
4. to develop some different types of educational software packages;
5. to develop software and tools for re-use in other projects;
6. to produce documentation[1] for use as examples;
7. to give an impetus to other Cluster I projects;
8. to generate ideas for other projects.

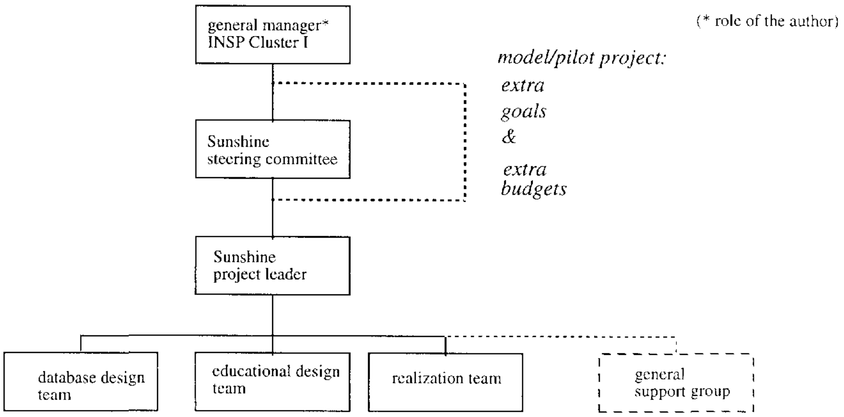


Figure 3.1 Organisation of the Sunshine model project. For the extra goals extra budgets were available. The general support group was also working for other (model) projects. In particular it provided services for the extra goals.

1. 20 technical reports about the development process and the product of this project were published by the Ministry. The first technical reports are summarized in HOEP (Hartemink, 1988). In this study most technical details of the Sunshine project are based on the contents of these technical reports sometimes without explicit referencing.

Although these were different goals, the main activity was the design and realization of an educational software package (goal 4). The internal organisation was structured in the manner depicted in fig. 3.1 and the steering committee was fully committed to the development of the educational software. The remainder of the goals listed above were controlled by the general manager.

The general manager of INSP Cluster I was responsible to the INSP organisation at the Ministry of Education and Science for this Sunshine project. The general manager was responsible for and controlled a number of other projects at the same time. The steering committee advised the general manager and controlled progress via the project leader. The steering committee consisted of members with expertise in subject matter, educational technology and modern media. The different disciplines represented in the steering committee were: a senior[1] multimedia specialist; a senior educational software specialist, a senior subject matter expert and the project leader.

The steering committee had the following tasks assigned beforehand:
1. to carry out a feasibility study for the project;
2. to advise on the formation of the project teams;
3. to supervise the progress of the project;
4. to approve the final reports of each phase;
5. to give advise to the project leader and members of the teams;
6. to improve the acceptance of the final product.

The project leader had the following tasks assigned before hand:
1. to oversee the correct application of the HOEP-method and the tool used;
2. to appoint the members of the project teams;
3. to organize the contribution of the general support group to the teams;
4. to report to the steering committee;
5. to control the budget;
6. to provide resources when necessary;
7. to provide "model" documentation;
8. to seek externally financed work for the realization team.

The phase leaders were responsible for co-operation within the teams and the achievement of the set goals and milestones.

With regard to staff availability the following was prescribed (all figures are averages):
1. availability of the project manager was not planned;
2. availability of the members of the steering committee was also not planned; however, 6-9 meetings were planned at the start of the project;
3. the project leader was available 1 day a week;
4. the phase leader of the application design team was available 3 days a week;
5. the phase leader of the database design team was available 1 day a week;
6. the phase leader of the realization team was available full-time;
7. availability of the other team members varied between 10% and 90%.

The database design team consisted of 4 persons with total hours equal to 0.7 full-time

1. Senior is taken to indicate at least three years experience.

equivalent (FTE); the application design team (including the varying contribution of the project support group) consisted of 6 persons with total hours equal to 2.7 FTE; the realization team consisted of 3 persons with total hours equal to 3.0 FTE (including some over-capacity).

To provide good communication the following formal means of communication were organized:

1. plenary meetings with all persons every 3 months;
2. phase leaders meetings with project leader every month;
3. the phase leaders organized frequent review meetings to improve standardization and communication between the teams.

The project leader wrote five progress reports of the project for external presentation. The minutes of the meetings of the steering committee were included with these progress reports. The Sunshine project was carried out using the following time schedule:

feasibility study:	3 weeks
specification phase:	30 weeks
realization phase:	50 weeks
acceptance test:	1 weeks

In the feasibility report milestones were defined and a budget outline was given. During the start-up phase the budgets for each phase were planned in detail. The total budget for the Sunshine project (for developing the educational software and for the extra work cause by it being a pilot project) was about 1 million DFL. The relative budgets for the various activities are listed in table 3.1. The internal organisation was evaluated and changed during the project.

activity	budget (%)
1. design team (database and application)	17
2. realization team	48
3. steering committee/ external support	3
4. project management	3
5. other media development	7
6. formative evaluation	2
7. testing in schools	3
8. final production	2
9. documentation and PR	4
10. administrative support	2
11. unforeseen (10%)	9
TOTAL	100%

Table 3.1: Budgets (relative) for the subsequent phases of the Sunshine project.

3.1.3 WHAT WAS DEVELOPED

The Sunshine project started as a pilot project with two main goals: one to solve an educational problem with specific subject matter and two, to provide a good method for development in other projects. “Weather and Climate” was chosen for the subject matter. This subject was considered to be suited to different school types, to different age groups, it can be taught and learnt from within the curriculum and without and it influences everybody’s everyday life. This subject has been used many times in the application of new technology. Considerable source material is available for example texts, tables, numerical data, graphics, and pictures.

The Steering Committee (SC) studied a survey of applications of this subject in curricula for different age groups within schools. After this preliminary study a "sourcebook" was produced containing a set of examples of instructional material on the subject for almost all student age groups. The SC proposed the development of a multimedia database with data for many applications in Weather and Climate. The applications had to be made for the first year of secondary schools. The SC advised "that not only ready-to-use educational software should be produced" but rather "creative use of new media in the schools should be stimulated".

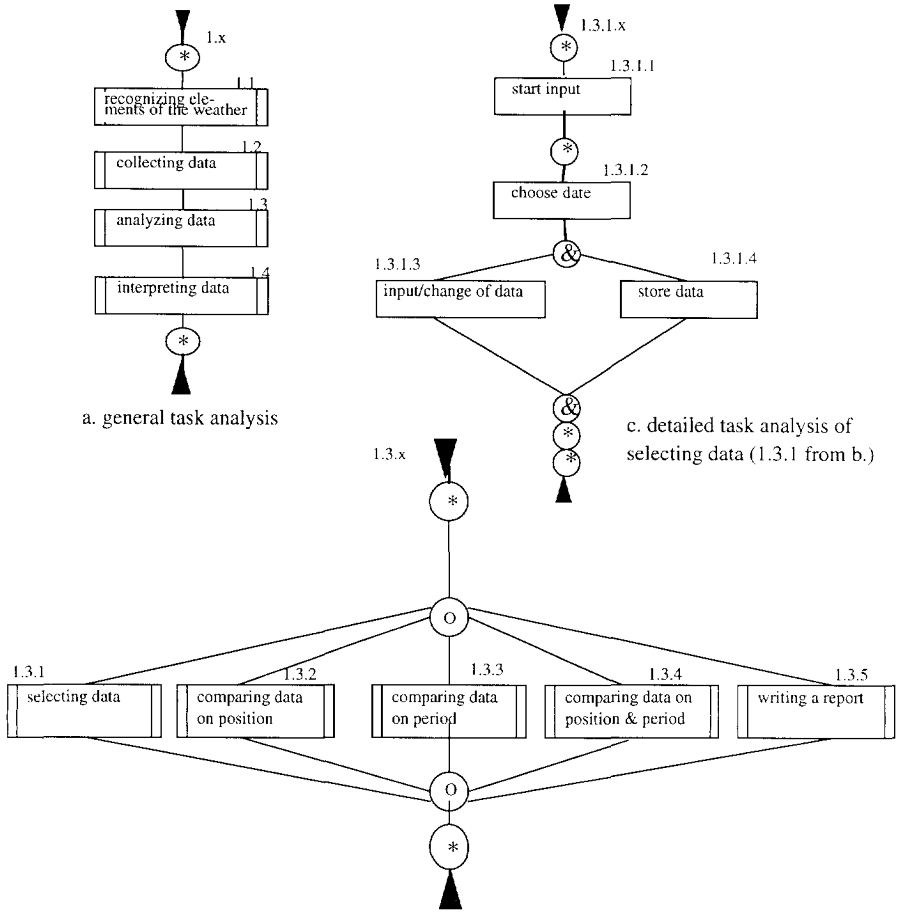
The database design team developed:
- a functional specification of the database;
- a logical design of the database;
- a general physical design of the database;
- a description of the requirements for the target computers in schools;
- a summary of the general supporting functions needed to maintain and use the database;
- a general description of the interface between the database software and the weather station application.

The database design team concluded (after much discussion) that it was not feasible to implement the database within the available time and budget. Given this negative result the original goal for the educational software (a multimedia database) was redefined.

The application design team developed a Weather Station. The design and the specification are described in detail in Stoffers and Terwindt (1988). The design was made within three months and during this period five alternate application proposals with different objectives for different target groups of students were worked out. After comparison and discussion the conceptual design of the weather station was chosen as the definite design. According to the newly chosen proposal the main goal of the course was: "to offer the student a means to describe, collect, measure and process weather phenomena, temperature, rain, humidity, wind and cloud data".

We will now describe the design of this Weather Station, the complete script can be found in (Stoffers and Terwindt, 1988). The script of the weather station consisted of:
1. General task analysis; the main objectives are analyzed and described, ideas for student tasks were collected and described. The main objectives were related to (see fig. 3.2a):
- recognizing and describing weather phenomena;
- collecting weather data;
- analyzing data;
- interpreting data.

Composition diagrams (Hartemink, 1988) were used to describe the sequence of (sub)tasks. The relationship between tasks can be: sequential, conditional, parallel. The characters <o>, <&> and <*> are used to indicate <conditional OR>, <unconditional parallel AND> and <repetition until some condition becomes true>. The box with double edges indicates a composed task worked out at a lower hierarchical level. Phillips et al. (1988) describe the same diagram for task analysis in dialogue design.
2. Detailed task analysis; the tasks described in the general task analysis were worked out into small steps of concrete tasks to be carried out by the students, e.g. choose a period on the calender; select a region of the world; ask for reports; display data using graphs on the screen, see figure 3.2b and 3.2c.
3. Description of the program functions. The design was translated into screen designs on paper and into diagrams and descriptions of the programming functions needed to implement the detailed task analysis. State transition diagrams were used to define all states, see Stoffers and Terwindt (1988).



a. general task analysis

c. detailed task analysis of selecting data (1.3.1 from b.)

b. detailed task analysis of "analysing data" (1.3 from a.)
1.3.2: with respect to position on the globe
1.3.3: with respect to period of the year
1.3.4: with respect to both position and period

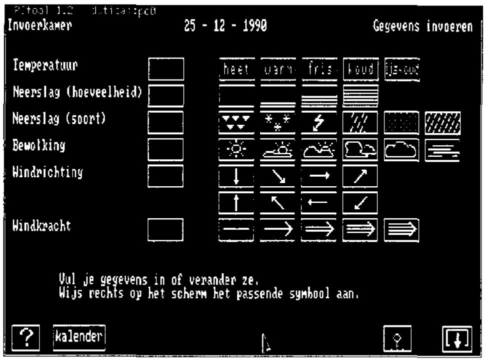
Figure 3.2 Task analysis of the main goal "Describing the weather" of the Weather Station.

The application design team gave a detailed description of the development process it followed (COI, 1987). It evaluated the way of working during the design phase with regard to the method, co-operation, contacts between disciplines and internal organisation.

The results of this internal evaluation were:

- the project was not based on requirements from real "problem owners". It seems to be difficult for educational technologists to recognize "educational problems" and to describe them. The designers did not like the term "problems";
- it is difficult for educational designers to think in terms of tasks instead of the program functions of the computer. They succeeded in finishing the task analysis down to the smallest subtask with encouragement from the project leader;

- the use of composition diagrams for documenting task analyses was received positively by educational technologists;



Invoerkamer 25 - 12 - 1990 Gegevens invoeren

Vul je gegevens in of verander ze.
Wijs rechts op het scherm het passende symbool aan.

Figure 3.3 . Screen of the fill-in form in the "input room" where task 1.3.1 is carried out.

- dataflow diagrams and state transition diagrams were good for media communication between the design and the realization teams;
- user interface design was underestimated by educational technologists because "no efficient prototyping tools were available" (the educational technologists were unable to prototype unaided);
- making abstractions from text to diagrams was difficult for the educational technologists. They need a colleague ("an educational software analyst") who is able to integrate skills of educational technologists and software engineers;
- team members from different disciplines sometimes lack the ability to listen to and explain things patiently to their colleagues;
- informal rather than formal contact between the design team and realization team is both necessary and sufficient.

The application design team also produced (with some external support) the user manuals for the teachers and students. The teachers manual gives a survey of the concepts behind the weather station, the learning objectives, the instructional tasks, how to introduce the exercises to the students and suggestions/ideas for classroom follow-up.

During the realization phase of the project the first version of the software and manuals were evaluated by subject specialists from an external service. In this evaluation a test run by teachers and students of the target age group at two schools was included. The result of the evaluation was positive and yielded a proposal for many small enhancements at the lexical and syntactic level of the user interface and some at the semantic level. Most of these proposals were implemented by the realization team before the final acceptance test took place.

The realization team first produced a quality assurance plan that described the characteristics, standards and tools to be used and contained chapters on project management, team organisation, programming environment, documentation standards and hardware requirements. The technical design was derived directly from the script and was composed of dataflow diagrams, state transition diagrams and pseudo code. All code was programmed in

ANSI-C and the Educalib library, see section 2.3.3. An installation guide and a maintenance manual were also included. A report on the acceptance test (after the formal evaluation was carried out) concluded the reports.

The final product, see figure 3.3, of the project was an educational software package called "Weather Station" with some written material for teacher and students. At this point the product was ready to go to a publisher for styling to the house brand and then publication, however, none of the Dutch publishers were interested in taking the risk, although the product would be provided free. The publishers were not confident of the market for educational software in the Netherlands. One of the main objections for them was that the educational software was not an extra to any commercial schoolbook series on the same subject matter.

3.1.4 EXTERNAL AUDITS OF THE PROJECT

As the project was a pilot the development process and the product were evaluated intentionally by an external expert[1] during and after completion of the project; internal organization and communication, task assignments and responsibilities, planning and budgets were all scrutinized. This is unusual, but was due to the pilot status of the project.

The internal organisation, see figure 3.1, was judged to be well chosen, implementation was far from optimal. The availability of project workers living at different locations (travelling times) was overestimated. Internal communication also proved to be organized too hierarchically and during the project informal contacts were increased to improve this. Task assignment was not performed well in all cases. For example the steering committee was also involved in some operational tasks and co-operation within the design team did not go well upon occasions.
The management of the project was not tight enough according to the external audit.

Goals (section 3.1.2)	Criteria evaluation/audit	Outcome
1. product implementation	no publishers interested	failed
2. improving the HOEP	HOEP was improved significantly	good
3. using development tools	only during realization, not during design	sufficient
4. evaluations	many reviews and formal evaluations	good
5. re-use	not re-used	failed
6. documentation	complete documentation published	good
7. impetus	only indirectly to some other INSP projects	sufficient
8. ideas	none within context of INSP	failed

Table 3.2: Achieved goals (Van den Camp, 1988). The outcome is a qualitative indication.

Responsibilities were not always met (caused by lack of time) or not clear (by lack of experienced staff). Planning and budgets were mutually inconsistent. The feasibility study was considered to be superficial. The first version of HOEP offered insufficient support for the application design team to work independently with it (the team was insufficiently trained to use the method well).

1. A senior consultant of a softwarehouse experienced in project management and design methods.

As a result of these problems the internal organization was changed with regard to frequency and agendas of meetings, project management, evaluation and review procedures; in particular the roles of the steering committee and the project leader were described explicitly in greater detail. These and other conclusions were reported in the series of technical reports on the Sunshine project and were used to prepare the second edition of the HOEP (Hartemink, 1988).

The eight goals (see section 3.1.2) were evaluated individually in the external audit. The outcome is summarized in table 3.2.

3.1.5 ANALYSIS

Introduction

In this section we will classify and comment on the development method of the project as described above in terms of Seligman et al. (1988). The project was especially interesting because one of the purposes was to evaluate and improve the HOEP development method resulting in release 3.0 (Hartemink, 1988). During the project extra resources became available to update the Handbook, thus, the documentation of the Sunshine project followed HOEP 3.0, this was only possible because it was a pilot project.

Way of thinking

Within the context of the project educational software was considered to be a type of discovery learning. Educational software was also seen as an integrated package of different media (software, written materials for teacher and student and classroom teaching by the teacher). To achieve discovery learning a student needs many degrees of freedom to find their way in the educational software. The Weather Station is a type of "open" learning system. In formal education students have often little motivation to learn, therefore one of the indirect goals of the project was to motivate the student by smart use of new media. The experience the student gains with the educational software complements the concepts, rules and examples offered in the classroom by the teacher. The written material for the teachers advised them on how to guide and stimulate the student at the individual level and gave some suggestions for group teaching. The educational software was intended for use in a computer classroom following a normal school time table. The student must fill in forms to prepare the input for their observations. The teacher can judge and review these forms with the students. At the end of the exercises the student has to produce a report which can be marked by the teacher. Thus, the teacher can control the progress of students by several means.

The philosophy behind the method is that developing educational software is a *technical* job that requires participation by experts in several disciplines (subject matter, educational technology, software engineering and project management)

Way of modelling

With regard to the complete life cycle of educational software the following model was used to determine who should participate in the life cycle (Van der Mast et al. 1991, pp. 357-358): "The first participants are the entrepreneurs, who are interested in the markets. The entrepreneur carries out market analysis and feasibility studies, and they decide to continue, they have to start up a project and form several teams to do all the work up to user testing. Then, they or their publishers arrive in the market arena to make a consumer product of the educational software, a difficult and expensive process. Testing in some schools and formative evaluation may take a full year because of the yearly cycles within any curriculum. Next, a school may decide to buy the educational software package after a proposal by the teaching team. In the best situation, this decision is made after careful evaluation of several similar packages and analysis of the costs of introducing and implementing them, in the curriculum. Then, individual teachers have to prepare, schedule, and plan use. The teacher must study many possibilities, including the tracking of student progress. Finally, the students will get the opportunity to follow the educational software at the moment they or their teachers think it is necessary. When using it, errors and ideas for improvement should be registered at the school. The publisher must arrange the logistics for collecting these comments to prepare a new release of the package.".

In the Sunshine project INSP management played the role of "problemowner". The life cycle of the product was continued upto the role of the publishers. No commercial publisher was found for important but not "functional" reasons. The publishers judged that there was no commercial market for the product. So, no commercial version was made of the Weather Station.

The overall modelling strategy consisted of the separation of the design of the educational contents and its technical realization in several media. The intermediary between both activities was the script. This describes completely all the static and dynamic attributes of the design. The realization of the design specified in the script by a realization team should be possible with minimal personal comments by members of the design team.

The realization started with a general and a technical design and then continued with programming this technical design. Yourdon modelling techniques were used for the technical designs.

The educational design was based on describing themes, concepts, objectives and learning tasks to achieve these objectives, the context was also analyzed. Then the design was continued for conceptual, semantic, syntactic and lexical design of the user interface. An overview is given of the modelling techniques used during the design process in table 3.3.

Components in script	Modelling/notation technique
subject matter concepts objectives	free text
learning tasks and subtasks	free text & composition diagrams
context study	dataflow diagrams
conceptual design	dataflow diagrams & text & tables
semantic design	tables & text
syntactic design	tables and state transition diagrams
lexical design	tables, text, data dictionary and pictures

Table 3.3: Modelling techniques during design of the Sunshine project.

Way of working

The way of working was, by definition, exactly according to HOEP and is described in section 2.2.7. The feasibility study was very important because the "proposed solution" was the result of solid educational analysis and design at the general level. One of the starting points was that this important first phase can only be done successfully by a team experienced in the disciplines educational software design and subject matter. For example according to HOEP one of the required sections of the feasibility report should propose and comment on:

- the required learning performance;
- some alternative interaction styles;
- an estimate of the expected complexity of response analysis and learning paths;
- a transformation of the general task analysis into the highest levels of composition diagrams;
- an estimate of the learning time needed by the students.

Only very experienced team members will be able to deliver this information.

The feasibility study of the Sunshine project can be considered to be a general quick reconnaissance of all design and realization phases to be encountered later in the project, both the proposed solution and the budget of the project were specified. The report contained the following:

- summary;
- introduction;
- analysis of the curriculum (problem definition, target group definition, subject matter analysis and instruction strategies);
- organisational and technical infrastructure (educational context, implementations context, technical infrastructure, dissemination and copyrights);
- proposed solution (instruction strategy, global content and structure of the proposed educational software package);
- financial feasibility and planning (cost/profit accounting, definition of project phases and budgets)

Way of controlling

In the HOEP method quality assurance (QA) is prescribed explicitly. After the decision to start the project as proposed in the feasibility report the first step is to make a plan and schedule the phases of the project and to produce a QA plan. The QA plan of the Sunshine project described commitments for human resources, for working environments for all teams and for secretarial and administrative support. Further the QA plan gave guidelines and "rules" to be followed during the project:

- how to adapt the phasing of the project during progress;
- the role and responsibilities of all team leaders;
- the way to register and control progress;
- the requirements for reports and documents;
- the planning of structured walk-through;
- some central meetings of all participants;
- assistance from a central support group for skill-training of the designers;
- other special support and the planning of the external audits on this project.

Beside this general QA plan a special QA plan was made for the technical phases of the project. In this plan, made by the realization team during the design phase of the project, details on the following aspects were analyzed and listed: specification of an MS-DOS/Unix network environment and how to manage this (classes of users, directory structures), portability from MS-DOS to Unix.

Way of supporting

Only two tools were used for the way of working and the way of control (project management): PC-Calc for project control, SCCS[1] on Unix for version control. During this project no real modelling tools were used for the educational design phase. Tables and texts were the main

1. Source Code Control System.

vehicles used to describe the design. The composition diagrams can be seen as clear overviews of what was presented in earlier text. The tools used to specify the composition diagrams were used to support the drawing, not the design of tasks. Commercial prototyping tools were used for some parts of the design in the early stages. During the technical design phases the IDE Software through Pictures system was used to specify a central data dictionary the elements of which were used in structure charts, dataflow diagrams and state transition diagrams. These modelling techniques were process oriented, as the learning process was considered to be a hierarchy of tasks to be done by the student. The library Educalib (section 2.3.3) was used for the programming phase.

3.1.6 CONCLUSIONS

Most goals of this pilot project were defined in such a way that it is almost impossible to conclude to what extent they were attained (Van den Camp, 1988). The Sunshine project had too many different goals to be considered as a representative development project for educational software, however, many aspects can be considered as examples. A very important product was the good documentation of all the activities during the project, from minutes of steering group meetings to detailed reports on the evaluation of the product, and, most documents were evaluated by the external auditors. These audits were published in the PMI-series of reports. In the final external audit (Van den Camp, 1988) the goal to produce "model"-documentation on the total development cycle was considered to be the one best achieved.

The educational software produced by this project was judged positively by an external evaluation in the classroom, however, the goal of use and implementation in the Dutch educational system failed. Dissemination and implementation were not planned and scheduled as part of the project. At the start of the project educational publishers were expected to be interested in "Weather Station". At the end of the project the publishers decided that the market for educational software in the Netherlands did not demand products like this. This is one of the built-in risks for "open" national stimulation projects where the free market is expected to adopt the products.

In Van der Mast (1989) we conclude from an evaluation of the activities of Cluster I of the INSP and among others of the Sunshine project ".. that it has proved that developing *reliable* educational software products is possible" and go on to state that the important question is "*what* products must be developed and *how* can they be implemented". In Van der Mast (1989) we also analyse the complete life cycle of educational software and conclude that the weak link lies between the partners in the educational market. This may be an important problem, but the conclusion that development of "reliable" educational software products is possible, is too simple, because general strategies for implementation and use must be taken into account as part of the development process. Thus, the Sunshine project shows that a product that is not used is not a "successful" product.

Incorporating strategies and methods for implementation is essential for any method for the development of educational software, as it is for the development of e.g. a new car.

Further it appeared that the way of modelling and realization in itself were not a problem, separation of design and realization also did not introduce problems. The way of controlling, however, appeared to introduce problems, difficulties with internal communication among the different disciplines in the design team inhibited smooth agreement on quality criteria. This resulted later, in difficulties during the acceptance procedures. The project leader was unable to improve the quality of the design directly because he was not a subject matter expert. He was able to improve the way of working of the designer team, but not the semantics of the design itself. Finally, it is clear from this project that the use of good supporting tools cannot replace "good" and experienced designers.

As a general conclusion from this inductive case study we can state that the aspects of modelling, working and supporting were covered and supported explicitly by the HOEP method. The aspect of thinking concerning the product was also worked out sufficiently but the way of thinking about the *use* of the product was ambivalent. Probably because Sunshine was a pilot project to improve the development process of many other development projects of the INSP *commitment* to prepare for the use of the final product was weak. The educational publishers in the Netherlands were not willing to take over the product because implementation and real use was not taken up with them from the beginning. The feasibility study (Van Beckum et al., 1987) fell short at this point. Referring to our framework we can conclude that the way of controlling failed at the very essential point of preparing for the *use* of the product. The way of controlling was also insufficient according to the external audit (Van den Camp, 1987). One of the main reasons given was that many team members, especially subject matter specialists and educational designers were *inexperienced* in developing educational software. This became most apparent during the *quality control* procedures at the end of the project.

3.2 The Teleac *PC-Privé* project

3.2.1 INTRODUCTION

The Stichting Teleac[1] at Utrecht provides education and information programs for adults using television, radio broadcasting and written materials.

A fixed number of hours is available from the Dutch public TV system for broadcasting each year. About 200,000 students per annum subscribe[2] and many more follow the courses offered by Teleac. The courses have an essentially multimedia approach centering around the medium of television. A blend of the following list of additional media is chosen for each course: radio-programs, printed material, devices and equipment for exercises, VCR, videodisk, CD-I, Teletext, educational software and commercial software packages.

The development method applied by Teleac was analyzed by the author by interviewing several senior project managers. The development and production of one course, *PC-Privé*, was analyzed in detail.

The organization at Teleac is very professional. Employees[3] experienced in all the necessary disciplines are available for course projects. *PC-Privé* had been broadcast once at the time this study took place (1992). According to the project co-ordinator interviewed, the development method used followed internal standards within Teleac and was quite representative for the project management of some 20-30 courses Teleac produces every year.

The subject matter for the course was MS-DOS and the fundamentals of the standard applications, text editing, desktop publishing, databases, speadsheets, book keeping and data communication. The potential audience for the course is anyone who starts using a PC at home without professional support and training.

According to the "media guide" the course consists of:

- two diskettes with software to be used during exercises. This software is not educational

1. Translation from Dutch: foundation Teleac.
2. i.e. buy the written materials from Teleac and bookstores.
3. The function names are modified from internal Teleac standards.

software according to our definition. It consists of educational (standardized user interface and simplified functionality) versions of a text editor, a desk top publishing program, a data base program, a spreadsheet program, a book keeping program and a datacom program;
- a textbook presenting the subject matter, exercises and the manuals for the software (Teleac, 1991);
- 8 TV lessons (broadcast once a week), also available on video cassette;
- 9 radio lessons (broadcast once a week), also available on audio cassette;
- a telephone counseling service.

PC-Privé is a course for self-study by a general audience. The contents of the various media are highly integrated. The goals are:
1. to provide a conceptual framework of meaningful tasks and actions that can be performed by a PC for private use;
2. to provide a survey of the standard software available for PC's in general;
3. to provide the opportunity to gain experience with using all these types of software on PC.

The textbook gives an introduction to the PC discussing the hardware and the function and use of the operating system MS-DOS (two chapters). It also presents, in six chapters, the different packages of standard software delivered with the textbook. Further it consists of manuals and exercises for these software packages. The eight TV lessons present the materials of the eight chapters of the textbook emphasizing the aspects best treated visually. Demonstrations of the course software are also given. During the period of broadcasting on TV students may call for assistance with understanding the subject matter and using the software. The radio programs present the most interesting problems encountered by the telephone teams and study hints are presented and discussed weekly.

The *PC-Privé* project was investigated here as an interesting example of the development of different media including software for use in an educational context. The software used is not in itself educational software but is the subject matter of the course. The contents of all media are highly integrated, e.g. in the textbook and the TV programs the use of the software is demonstrated directly. The software was made specially for this course. So, that the smallest details of the user interface of the software would be exactly the same during the demonstrations on TV, in the textbook and during the exercises carried out by the students on their PC.

The method for developing the *PC-Privé* course is analyzed here in terms of the framework presented in chapter 2.

3.2.2 ANALYSIS

Way of thinking
At the beginning of a new project the management board of Teleac performs a feasibility study on the subject matter and the perspective in the market of the new course. This study is done by a few senior internal staff members. When the outcome is positive a project team is appointed with internal staff for each main discipline represented in the product. The project team is chaired by a senior project leader who chairs 2-3 projects simultaneously. The team is responsible for the quality of the course and for delivery of the product within the time constraints and the budget. The team is small (3-4 persons), highly motivated, and focussed strongly on to the quality of the overall product, permanently tuned to the target group and the market, having excellent expertise in all media involved.

The team makes all specifications of content and form of the components of the course. It hires specialists for development and production of components when needed. The responsibilities of all team members are described exactly beforehand. These *responsibilities* and not the intermediate design products are the basis for the organization and scheduling of the project. All team members, most of the time, work on a few projects at the same time. The broadcasting schedule strongly stipulates the planning of the development.

Way of modelling
The way of modelling is based upon documentation, hierarchies and sequences of components. All documents consist of texts and labels within. "Objects" of the subject matter are described and the tasks/procedures which must be performed by students. The content is described and structured once and used for the design of all the media. The number of TV programs was, for practical reasons, defined very early on as 8, presenting one cluster of subjects each. This structure is also applied to the chapters of the written and other media. The content is specified by the project team in terms of subjects, objectives, aspects to be emphasized and examples for each chapter. After this, authors who are specialists on the relevant subjects are hired to write the textbook according to the specifications. At the same time the design team produced a complete scenario/script for all TV programs. In this script scenes and texts to be spoken are specified along a time line using seconds. At the same time the requirements for the software package are also derived and postulated. The assignment to develop the software was given to an external company. The software demonstrations to be presented during the TV broadcasts were prototyped, exactly, at an early stage of design. These prototype demonstrations were recorded using professional TV equipment to define the exact form and instructional quality. After definition of form and content of these demonstrations the definite requirements for the written manual for exercises by students were specified for the external professional author.

Way of working
The project team consisted of the following actors:
1. project co-ordinator;
2. TV director;
3. radio director;
4. publisher of written materials;
5. internal support from Teleac, when needed, for:
    - research
    - public relations
    - student support

The activities during the project were ordered in a non-standard way for Teleac, see table 3.4. The project was launched after the decision of the feasibility study to "go". The project team was assigned and the project started officially. First, "research" was done on the subject matter and the educational requirements. Possible external subject matter experts were contacted for writing material. The educational design of the whole course was completed within three months by the project team. The educational design consisted of a textual description of all objectives.

After approval of the design a plan was made for writing and producing the course book[1]

1. The course book consists of a textbook and the exercises and manuals for the software. The textbook and the manuals were produced by different authors and integrated by one editor.

and research was done on standard software packages, this resulted in a set of definite requirements, the assignment was then given to an external partner to design and realize the software. The writing process of the textbook and the development of the TV scripts by the TV director were done lesson by lesson during the same time period. The script drafts for the lessons were then passed to the TV director.

The software manuals and exercises were written simultaneously. To meet the time schedule an extra author was appointed. The TV programs were shot in a period of about 6 month during the final editing phase of the written materials. The first broadcast occurred immediately after the TV programs were produced.

PC-Privé project activities
market analysis and feasibility study followed by a decision to continue this project
appointment of the project team
research on subject matter and looking for expertise
first general educational design
general design of the textbook
research on requirements for software (subject matter of the course)
requirements for the software
design and realization of software
first version of textbook studied by TV director
development of TV script
contacts Postbank Girotel
development marketing materials
writing software manuals
appointment of the editor of the textbook
revision text book
assignment extra writer/author
assignment new author for manuals/exercises
writing manuals
design radio programs
design and organisation counselling service
TV filming
radio production
broadcasting TV and radio

Table 3.4: Way of working. The following phases were started in this sequence and performed partly simultaneously.

Way of controlling

The project teams of Teleac are part of a matrix organization. Most team members are involved in more than one project at a time. When the project team is formed the budget and the data for broadcasting are also well advanced. Quality assurance is assured by the appointment of very experienced senior staff, who have a broad overview, and are very conscious of the properties of the target group. The members of the team are able to fully understand the content of the course at the level of the students and they are able to apply their knowledge of the medium both during the design phase and during the testing of products made externally by others. Review, correction and improvement of the final products is done very intensively in co-operation with external partners under the strong supervision of the member of the project team responsible for this part of the work. All members of the project team are used to working very analytically and critically and they are able to perform acceptance tests for all components of the course "in the role of the student". The separation of phases is weak, co-operation within the team is very close and many iterations are performed, however, as said earlier, the responsibilities of each member of the team are described and agreed upon very clearly and strongly. Most projects take 2 years, though the *PC-Privé* project was shorter, about one year. The product was judged to be successful and the course had been broadcast four times by the end of 1993. According to the management of Teleac this was partly due to the quality of the organizational and motivational skills of the team.

Way of supporting
Modelling and working at Teleac are not supported by explicit methods and tools. Only the controlling of the budgets and project management was done using standard spreadsheet software.

3.2.3 CONCLUSIONS

The development method for educational products at Teleac is characterized by a strong concentration of excellent craftsmanship in many disciplines and media into a small team. This team is expected to design, develop and evaluate successfully products on their own or by the supervision of hired external specialists. Much emphasis is put on the personal responsibility of each member for well defined parts of the process or product. It is particularly striking that the team has so much expertise on the market and on student behavior that continuing formal or external evaluation of the product by subject matter experts was not necessary. This is a risky way to operate in an open consumers market. Effective quality assurance is possible at Teleac by having good knowledge within the team about the target groups and by co-operating closely in a small team.

The greatest difference between the Teleac course and standard educational software is that in the school situation, as with on the job training, the student is a well known partner in the teaching and learning process who can be tested and interviewed. It would be interesting to investigate the way Teleac develops courses that prepare students for examinations.

This project has in common with educational software projects that the integration of expertise from several disciplines is needed for success. This project differs, however, from educational software projects in the absence of support for the personal instructional dialogue. All educational problems encountered and solved here are at the macro (organization) level and the meso (presentation) level. The micro level of structured individual interaction is not considered.

It can be concluded from this project that integration of different disciplines and expertise at the macro level is best guaranteed by team members who have a broad overview and a lot of experience with different disciplines and hence are able to provide the necessary integration within their mind and by working implicitly.

To develop educational software far more analytic and modelling work has to be done to produce a general storyboard and a detailed script. Often all these analytic, explicit and creative steps are difficult for one person. Yet, we may conclude from this project that it is advantageous, for developing educational software, to concentrate as much as possible expertise in a very small team that can control the quality of the design internally and manage the work of externally hired experts efficiently.

We can conclude from this interesting inductive case that describing responsibilities of all team members accurately and requiring commitment have to be an essential part of the ways of working and controlling. The assignment of one director who is responsible for and able to control the quality of the product under design continuously is also important. Communication between team members can be improved when they are experienced in more than one discipline. This can improve the way of working. Separation of disciplines can be realized to a high degree by hiring external professional specialists. Quality control has to be supervised and realized by a highly motivated director. The way of modelling chosen in this project is of course based on the tradition of TV using storyboards and scripts. Finally, the media approach benefits the degree of engagement the student will show with the educational product.

4. TOWARDS A NEW THEORY OF EDUCATIONAL SOFTWARE DEVELOPMENT

In this chapter we present a new theory about developing educational software and criteria for evaluating this theory. We will use the framework of Seligmann, Wijers and Sol (1989) to present the theory. We summarize the relevant conclusions of chapters 2 and 3 for each aspect (sections 4.1-4.5) of the framework and then present a synthesis of concepts and procedures which form part of the theory for an aspect. In section 4.6 we present lessons from film making. Finally we derive 8 research questions for testing the theory in the case studies of chapters 5-7. This theory is evaluated and improved in chapter 8.

4.1 Way of thinking

In chapter 2 we concluded that the way of thinking in most methods except one, was neglected. None of the tools described supported really a way of thinking. In the *Sunshine* project (section 3.1) the way of thinking about the instructional problems and solutions was clear; but the project was not clear about the *use* of the product. In the *PC-Privé* project (3.2) the way of thinking on the product and its use was very explicit. Departing from the conclusions in chapters 2 and 3 we will now apply information systems development standard questions, why, what, and how to the way of thinking of developing educational software.

Before development of educational software can start the question *why* it is needed should be answered. To assure complete implementation and use within an organisation or school it is necessary to convince all managing and participating parties that this question has been answered sufficiently. This persuasion will stimulate their commitment to contribute to the costs of producing educational software, cf the *PC Privé* project analyzed in 3.2.

In educational software the subject matter represents the functional component defining *what* is to be learned. Didactic and educational strategies and interaction style represent *how* the subject matter is to be learned. Both are equally important for the success of this communication medium. With educational software, the greatest problem is not to specify the subject matter at the higher and the detailed levels as it is often well described in text books. Sometimes, for training purposes it may be difficult to analyze carefully how a task (e.g. selling a product) is performed to specify exactly what training is needed. In our opinion, the most important problem is the integration of different components of the *how*: the instructional strategies for the finest details of the learning dialogue, the "look and feel" and the user interface, at different levels of abstraction, at the functional level of "tasks" a student must perform, and at the lower level of the interaction between the learner and computer.

With conventional information systems development, see figure 2.2, functional design is very important to specify the practical goals and effects. The user interface and interaction are important but not significant factors of the design and the product. With educational software as a component of an educational information system both the functional design and the user interface are essential for the characteristics and the quality of a design. Functionality and interaction must be integrated in a way that determines the quality of the system. This can be compared to products with an artistic component where the semantics, message or content, and the art of presentation are both essential for the quality and success of a product (e.g. film, opera, ballet). To describe this difference between educational information systems and conventional information systems, we will introduce the micro, meso, and macro levels of an information system and apply them to educational software.

We can now work out the why, what, and how questions using the IS/RS paradigm of figure 1.1 starting with the level of the learning unit. Interaction between learner and computer or computer controlled device is the most essential component of the "medium" educational software. This level is called the *micro level.*

Despite the automated interaction, the role of a teacher/instructor is always there: very present in the classroom or in the far background with distance learning. The presence of the teacher may be very important for the successful use of educational software. Therefore their role should be considered explicitly during the design phase. The teacher "manages" the individual learning process of a student at the level of the teaching task. This level is called the *meso level.*

The school or the organisation where training takes place provides the general curriculum, the facilities infrastructure and the logistics for the instruction process in which both student and teacher are involved. This is called the *macro level.*

These levels are not independent and isolated. A characteristic of educational software is the necessity to integrate design aspects at the micro level with aspects at the other levels. In chapter 1 the IS/RS paradigm is introduced to depict an educational system. The "learning unit" in figure 1.1 represents the micro level and the teaching unit the meso level. The macro level is positioned at a recursively higher level of abstraction.

Our way of thinking requires that all three levels are important during development of educational software but most emphasis has to be put on the micro level where the medium aspects are found. The individual learning process is the principal goal the designers must have in mind. At least one member of the design team should understand both the target group of students and the content material profoundly and be able to test and evaluate the results of the ongoing design with respect to the characteristics of the target group. Learning is the principal goal, interaction is one of the facilitating conditions for achieving this. To achieve maximum "quality" of educational software the best creative designers are necessary to design the "how". The demands can be compared with success variables in the entertainment industry rather than in teaching at ordinary schools. One of the demands is to implement quality assurance methods during the development process.

In this framework educational software is considered to be a strategic product for the student and the organization. The highest professional quality is demanded of the final product. Therefore the development process has to be organized in an as small as possible multi-disciplinary team with highly skilled and motivated members. The team must be able to understand and to evaluate all the details of the learning and interaction processes. The members need a strong image of the many aspects and overall characteristics of the design. Educational software should be considered to be a product of fiction/imagination based on and manufactured with craftsmanship and professional skill. Within the development team co-operation and mutual communication between specialists from different disciplines are necessary. The following specialisms should be available within a team assigned to develop a strategic multimedia product. For a market oriented product the first specialism (general management of the organisation) can be replaced by a person who knows the market very well. An indication is given for each specialism of what question(s) provide the focus.

1. general management of the organisation (why);

2. project management (why, what and how);
3. subject matter expertise (why and what);
4. instructional specialism (why and how);
5. software engineering (what and how);
6. media specialism (how).

The most crucial aspect of these specialisms is the fulfillment of the role of principal project manager. Heines (1984) looked for role models in screen design (user interface design) for educational software. He proposed that the best role model would perhaps be a combination of graphic art and "informational television", news programs, documentaries, sporting events, children's educational programs. Heines states that "the screen designer can learn important concepts and borrow techniques". This role model can be extended beyond concepts and techniques for screen design of educational software. The principles of communication are universal (Heckel, 1991).

This holds for all communication crafts e.g. writing, advertising, journalism, film making, choreography, musical composition and teaching. Heckel describes the transition of filmmaking from an engineering discipline to an art form. "Movies did not flourish until the engineers lost control to artists, or more precisely, to the communications craftsmen". The perspective of the engineer is *what* the system does, the artist's is *how* the system should do it. Heckel describes how movies are designed and presents this as an example for software design. The design of a movie starts with a story written in text. This story is presented visually through story sketches. "The story sketch should show character, attitude, feelings, entertainment, expression, type of action, as well as telling the story of what is happening." When the story sketch is approved many components are added to create a prototype (story reel) of the design. Everyone from all disciplines of the team has to study and revise this. In this way a medium is provided to make it always possible "to get a good feeling for the current "best guess" of what the audience's experience would be".

Today, user interface designers are also trying to integrate "communication" and "engagement" skills into their expertise (Clanton et al., 1992; Young and Clanton, 1993; Smets et al., 1994; Jacques et al., 1995; Van Aalst et al., 1995a; Van Aalst et al. 1995b).

The concepts of script and scenario are well known for educational software design. A script is an accurate functional design (Sodoyer, 1990) and a scenario is a brief summary of the scene and story of a communication product, a context for improvisation by all disciplines involved (Heckel, 1991). In addition to this, the role of director of a film or a musical can be added to the comprehensive model for the way of thinking. The director is responsible strongly for the quality and the success of the result of the project. He "owns" the initial concepts and believes in them. He searches for the best designers, artists and interpreters. He personally guides and controls all members of the development team. Finally, he has to evaluate the quality of the design and the partial products continuously, not only at the conceptual level but also at the levels where technical skills (camera work, stage composition, music, choreography, etc.) are essential. The project manager should have at his disposal enough resources to control and guarantee the quality after all the evaluations he makes. This role of "director" has also been described for choreographers by Van Manen (1992) and Calvert et al. (1993). Laurel (1991) describes the design of interactive media including educational software and entertainment multimedia products from the metaphor[1] of "theatre", see also Frenkel (1994). Gimpel (1992) describes the designing and building of the medieval cathedrals, the necessary collaboration between many disciplines and the management of the development process (concurrent engineering). A cathedral can be considered as an object that "interacts" with people (Van den Berg, 1969).

The approaches to the way of thinking discussed here are supported by the role of the project manager in the *PC Privé* case analyzed in chapter 3. The project manager of the *PC Privé* case behaved as a director of a TV program in a way similar to that described above.

Finally, we can conclude, concerning the way of thinking, that educational software has to be considered as a medium and that emphasis on instructional and technology aspects should be balanced carefully.

4.2 Way of modelling

In chapter 2 we concluded that the way of modelling in all methods and tools can be characterized as addressing either subject matter specification, or instructional design, or technical design. Modelling user interface design is neglected. When considering educational software as a medium, modelling the user interface must be an essential part of the way of modelling in general. Modelling the user interface requires modelling actions and tasks the user can do.

Modelling requires techniques to describe knowledge about the concepts of tasks the user/ learner can do and user interface objects. With regard to developing educational information systems, concepts can be viewed at different levels. Working out these levels, distinguished in 4.1 for the way of thinking, can be seen as an important aspect of modelling. The three levels macro - meso - micro depicted in figure 4.1 each require different modelling techniques.

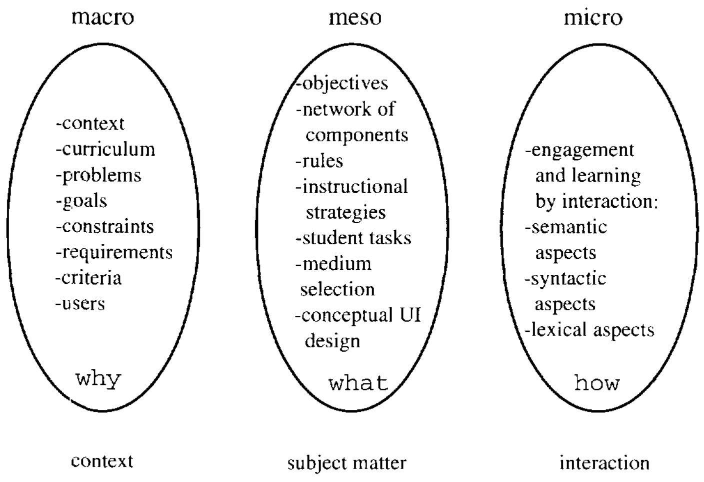


Figure 4.1 The three levels for modelling educational systems.

At the *macro* level (see figure 4.1) the general requirements and the teaching and learning goals can be derived from the organizational context and the given constraints for the product. At this level models can be used which are not formal and accurate but are flexible and subtle. The

1. ,The meaning of the word metaphor is not "user interface metaphor" but "a fundamental understanding of what is going on in human-computer interaction", (Laurel, 1993, p. 19).

results of this modelling are a description of the subject matter to be taught or learnt, the main learning goals and the performance criteria for learning by students. This modelling can be done verbally with some supplementary drawings and pictures. It is sufficient to list the subject matter in terms of subject, learning goals and criteria for measuring learning performance of the goals.

The pedagogic and didactic modelling is done at the *meso* level, see figure 4.1. This is the level of the pedagogists and the teacher's expertise. The results of the macro level are the inputs for this detailed modelling. At the meso level the terms of content components, goals, objectives, rules, instructional strategies, student tasks and medium definitions are used. Dataflow diagrams can be used to define the processes. Task composition diagrams can be used to specify possible student behavior. The building blocks of Godfrey and Sterling (1982) can be used as instruments to assist the modelling. A preliminary conceptual design (Wilson and Rosenberg, 1988, p. 864; Foley at al., 1990, p. 394) of the UI must be part of the modelling at this meso level. Transference of didactic thought is the especial aim of the meso level.

The macro and meso levels of modelling aim at abstract concepts that result in a specification of subject matter, instructional strategies and tasks, media selection and finally the conceptual user interface design.

Next, a design step must be made to specify interaction with the student. This is the *micro* level, see figure 4.1, where the static representation and the dynamic interaction of the user interface must be designed in detail. The conceptual modelling for the user interface has to be done at the meso level because the concepts are influenced strongly by the characteristics of subject matter and student tasks. At this micro level the user interface design has to be continued in conjunction with the semantic, syntactic, and lexical steps, see section 1.5. The result must provide the specification for the efforts of the realization team. A media approach is essential at the user interface level of modelling, as mentioned in 4.1.

At the micro level the conceptual design, roughly designed already at the meso level when the instructional strategies and the user tasks have been modelled, has first to be completed in a definite form. First of all the type of user/learner has to be known for the conceptual design of the user interface. Not their precise knowledge of the subject matter but more in general their level of education and the style they are used to for learning and working; and the concepts they are used to for communicating about the subject matter. Large differences can be expected for age and education of children or adults, for customs and habits in different sections of business or schools or ethnic backgrounds. The conceptual design should match the "character" of the user. The conceptual design can be specified using text and illustrations (story boards) of interactions. The semantic design represents the functional design of the user interface. The contents and the content independent elements of the interaction, for example presentation of objectives, hints and feedback to the learner, must be integrated into the didactic design strategy.

The syntactic design can be modelled using text and state transition diagrams to describe the dynamics of the interaction.

The lexical design describes the static components of the interaction, the layout, and can be modelled using text and detailed illustrations of screen layouts.

Modelling at the macro, meso and micro levels must be performed by people with experience in different disciplines. Ideally, one person, the project leader/director, has an overview of work at all levels and she is able to evaluate progress at any moment. She must be able to see, at all times, the consequences of design decisions for interaction and learning by the student. This has consequences for the way of controlling. Modelling at these three levels must be integrated into the way of working.

The specifications produced during the macro, meso, and micro levels must be the starting point for system design and technical design. The technical modelling and design is not domain dependent and can be carried out using conventional software engineering techniques for highly interactive software, e.g. Hix and Hartson (1993). This is not worked out here.

An important question is how the macro, meso and micro levels of modelling can be related to the why, what and how questions to be answered during information systems design. At the macro level emphasis is put on answering the *why* questions, see figure 4.1. The context of the curriculum, educational and organisational problems and requirements, users and performance criteria are the matters of concern. At the meso level emphasis is put on the *what* questions. The what questions concern the contents of the educational software: the subject matter, the network of components, instructional strategies, objectives and learning tasks, medium selection and conceptual design of the user interface. At the micro level the emphasis is put on the *how* questions. The how concerns motivation, engagement and learning by interaction. Semantic, syntactic and lexical aspects have to be considered. The how questions are, however, not the technical how questions of conventional information systems design. The crucial place of interaction in educational software needs a distinct design level for answering questions of how the details of the interaction should be designed. The technical "how" questions must be put and answered during technical design.

Finally, we can conclude concerning the way of modelling that for developing educational software explicit models are needed for the macro, meso, micro levels specified here. Several modelling techniques are available from the survey in chapter 2 but it is not satisfactory to apply one isolated technique for one (aspect of a) level. For the micro level professional modelling techniques should be added from the discipline of computer-human interaction (CHI). The *media* way of thinking, see 4.1, is mostly needed at the micro level of modelling.

4.3 Way of working

As discussed in chapter 2 the way of working is described in terms of the developer's tasks, their definition, their parallelism, sequence, iterations and synchronization. Many tasks of the developing team can be done in parallel. To improve the efficiency[1] of the overall development process some order is needed for answering the questions why, what and how. It is also advantageous to agree upon important aspects of a design before realization of these aspects is initialized. The three levels of modelling of section 4.2 give rise to the separation of the way of working into rather independent phases because of the differences in the disciplines and skills needed for designing at each level, see figure 4.2.

This way of working is an enhancement and a refinement of the HOEP and the PRINT methods described in chapter 2, figure 2.3 and tables 2.1 and 2.2. The macro, meso, and micro levels of modelling, see figure 4.1, can be carried out during explicit phases each using people with specific disciplines and skills. These phases have to be distinguished explicitly for the way of working.

A feasibility study is considered to be a pre-project activity before the budget of the project is known. During the feasibility study all the relevant disciplines and expertise are needed to make a good "guess" in a short time, of the identity of the problem and the feasible solution.

1. If time and investment are available (e.g. many strategic products demanding extreme high quality) it is of course possible to arrange a contest of more than one design team working independently on the same requirements. This is analogous to architecture design contests.

The manager[1] of the whole project (all phases) should be highly involved in thinking about the solutions to bring in their expertise and to promote total commitment to carrying out the project if a go-decision is made. After a managerial start-up phase the design has to be carried out in three steps with several iterations. The growing "idea" of the design must be passed as a baton through the macro, meso and micro levels using prototypes[2].

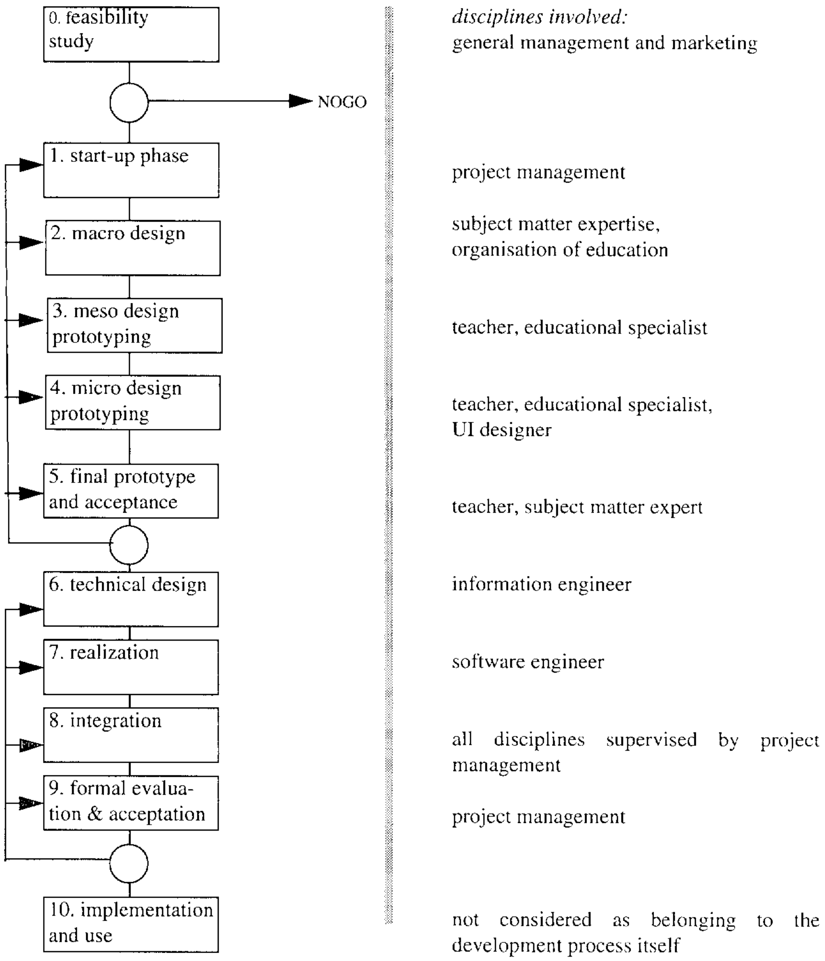

0. feasibility study
NOGO
1. start-up phase
2. macro design
3. meso design prototyping
4. micro design prototyping
5. final prototype and acceptance
6. technical design
7. realization
8. integration
9. formal evaluation & acceptation
10. implementation and use
disciplines involved:
general management and marketing
project management
subject matter expertise, organisation of education
teacher, educational specialist
teacher, educational specialist, UI designer
teacher, subject matter expert
information engineer
software engineer
all disciplines supervised by project management
project management
not considered as belonging to the development process itself

Figure 4.2 Overview of the way of working and the disciplines involved in each phase.

The final prototype and the specification documents belonging to it must be accepted before the realization phase. The realization phase should be carried out by professional employees

1. It is assumed in this new theory that a project of educational software development cannot be managed by an "independent outsider" who is not familiar with the subject and hence is not able to carry out quality evaluation personally (cf. role of film director).
2. Prototypes can be made with paper and pencil and with software packages for rapid prototyping.

using conventional techniques for software engineering and engineering of the user interface and other media.

phase	description of design activity	level	sme	edu	media	se	pm
0, 1	selection of the educational problem and definition of requirements for the solution	macro	x				x
2	analysis of the educational context and the subject matter. Definition of principal goals and how to test the achievement	macro	x	x			
0	feasibility study of alternative solutions by prototyping	macro+ meso	x	x	x	x	x
1	project start-up and QA-plan						x
2	finishing macro design	macro	x				
3	pedagogical design - prerequisites- network of goals and objectives- specification of rules and components- didactic strategies- definition of learning tasks (learning is process-oriented)- specification and sequencing of tests	meso+ micro	x	x			
3	media selection and design	meso+ micro	x	x	x		
3	conceptual design of interaction	meso+ micro	x	x	x	x	
4	simulation design (if a simulation is part of the educational software)	meso+ micro	x		x	x	
4	functional design including detailed design of interaction in semantic, syntactic and lexical aspects	micro			x	x	
6	technical design	micro				x	
7, 8	realization and integration of all media	micro			x	x	
9	review, improving and acceptation procedures	micro+ meso+ macro	x	x	x	x	x
9, 10	operational system delivery	macro					x

Table 4.1: Way of working: disciplines necessary for distinct design activities (not necessarily in this order). An x means that the discipline is involved explicitly in the design activity. Legend: *sme*: subject matter expertise; *edu*: educational technology expertise; *media*: media technology expertise; *se*: software engineering expertise; *pm*: project management expertise. *Phase* refers to figure 4.2.

To achieve professional quality it is recommended strongly to carry out the realization using skilled professionals. Although it seems "easy" to realize e.g. screen layout and to generate code with some modern tools, this can only be achieved by professionals who know the tools and techniques behind them very well. The realization team should be able to read and interpret the design documents and the prototypes. The general manager should be concerned with guaranteeing that the ideas of the solution stay unchanged over the phases and are past to new teams and without distortions with respect to the original and accepted ideas.

Prototyping is a way of working to improve the quality of a design. While developing educational software, prototyping should be applied at all the macro, meso and micro levels.

At the macro and meso level prototyping should be used to analyse educational problems and to generate ideas for solutions. At the micro level prototyping is necessary because educational software cannot be designed without intensive input from prospective students, and teachers if applicable. This input is needed to evaluate alternative solutions for interaction and didactic strategies. The results of the design at the three levels can be joined and adjusted by prototyping. While prototyping is necessary to obtain a good design the next step is to realize an operational product with the same characteristics (Sodoyer, 1990). Therefore, the responsibilities of the teams carrying out the phases depicted in figure 4.2, simultaneously or not, should be described accurately, the requirements for communication between the teams should also be prescribed.

The disciplines needed for different design activities are listed in table 4.1. This is a general list applicable to many subjects and styles of educational software providing multimedia components. Some design activities can be omitted for simple educational software products, e.g. textual drill and practice exercises. The contents of table 4.1 can be used as the starting point for a quality assurance plan, see 4.4.

Finally, we can conclude that the way of working with phases, disciplines and the skills required depicted in figure 4.2 and table 4.1 provides the outline of a project management method and a starting point for a quality assurance plan. The demarcation of the skills provides a means to specify responsibilities and requirements for communication within a team. Design (phases 1-5) should be separated, in different teams and at a different time, from realization (phases 6-7) because very different disciplines are involved. The necessary minimum communication should supervised by the project manager.

4.4 Way of controlling

In chapter 2 (see table 2.4) we concluded that the way of controlling is neglected in most methods described. In the *Sunshine* project the way of controlling also caused problems at the end of the project. Only in the *PC Privé* project was the way of controlling supported satisfactorily. The way of working and the way of controlling in this project were implemented using a style that is common in the world of TV and multimedia productions.

Controlling the development process based on the chosen ways of modelling and working, requires methods for planning and evaluating. To manage the employment and allocation of generally limited resources (Wijers, 1991) the phases of the development process must be documented and structured carefully. The way of working should be structured congruently with the way of modelling that is applied. During the development process this congruence should be controlled at clearly defined moments. The responsibility for controlling and evaluating the quality of the product should be clearly designated to one person. This person should be very committed and ambitious to guarantee product quality. They should have the power and the skill to manage all expertise they feel is necessary. It is essential that they can manage team members of all disciplines involved, as does the producer[1] of a musical or film.

1. Person in charge of financing and scheduling a film production.

This person should be able to imagine how the student will use the educational software at the micro level.

Quality assurance methods used as part of a well known project management technique, should be used to strive for completeness and clearness. Planning of budgets, resources, phases, documents, reviews and time schedules (and how to adjust them) must be provided. Human resources especially, have to be planned carefully because availability of the right expertise from the required disciplines is very critical for the progress of development according to the planning.

Finally, we can conclude that controlling plays a critical role in guaranteeing the outcome of a project. The central role of the project manager as a ubiquitous quality controller and the presence of a quality assurance plan are most important.

4.5 Way of supporting

From chapter 2 we can conclude that the commercially available tools provide support that is limited to prototyping and/or programming. The providers of these tools claim that they can be used by educational designers and subject matter experts, but these claims cannot be proved except for small one-person projects.

Many other tools and techniques may be used by each of the disciplines to support the development process in which several rather different disciplines participate. Each discipline has its own way of supporting based on either verbal descriptions, diagrams and formats or electronic prototyping and visualization tools. All can be applied in the usual way. The most critical needs are those concepts and tools to support communication *between* disciplines. It is assumed that no artificial means have been developed that have been proved to support collaboration between individuals from different disciplines[1] (other than annotation systems and email). Verbal and textual explanations augmented by pictures and graphics are necessary and sufficient. As the story reel is a medium to support communication between disciplines within the film making team, so electronic prototypes can play this role for educational software development.

4.6 Lessons from film making and TV

As mentioned in previous sections it is plausible to enlarge the prescriptive role of directing a film production to the comprehensive analogy of the complete life cycle of entertainment products such as film, theatre, ballet and musical. These are typical "communication" products (Heckel, 1991). The life cycle passes through:
- conception of idea;
- collecting funds;
- forming teams;
- writing proposals, scripts and storyboards (and approving them);
- analyzing end estimating markets and risks;
- making and evaluating prototypes and mock-ups;
- realizing and producing final productions;
- publishing, performing and issuing them;
- marketing and selling the final products;
- reviewing by public and professional media;

1. Computer Supported Co-operative Work (CSCW) is a new area focusing on facilitating communication between people at the same time/place or not. The benefits of CSCW for creative multi-disciplinary design teams have not yet been proven (Shneiderman, 1992).

- achieving commercial success or not.

Heckel (1991) quotes Dave Hand[1] when he says that the raw material for any communication craft is what is in the audience's mind:

"Our entire medium is transference of thought. The thought is created first in the mind of the storyman... then transferred to the director, who attempts to transfer it to the animator... The animator then attempts to transfer it pictorially. He takes it out of the intangible, and places it in tangible form, in picture, for transference back to the mind of the audience... and picture presentation is clearer than any other means of transferring thought from one person to another."

Heckel describes (Heckel, *1991* p. 180-184) how a typical Disney feature film would take three and a half years to make:
Six months of research, one year of work on story, styling and experimenting; one and a half years of animation; and six months of follow-up to add color and music and to photograph the 460,000 drawings. The process starts with a *story sketch*[2]. Story boards are springboards for new and better ideas at story meetings. Final versions serve as useful prototypes for the final film. When a sketch is approved the dialogue is added, but the story boards lack the element of time. A film is made for protoyping the rough sketches including a rough sound track. This is called the *story reel*. Everybody on the team must study it: the layout people, the animator, the composer, the storyman. Each person should be able to see how their part fits into the whole. All revisions are immediately inserted into the story reel which slowly changes into the finished film. A final Disney film will consist of only 20% of all drawings made to develop it. This approach is in sharp contrast to the other animation studios of the thirties. Disney created a set of tools that helped him make a film: some to help make a quality finished product; other management tools were designed to keep track of the process and ensure that all the details merged together correctly. "The main focus of these tools was to make it always possible for everyone to get a good feeling for the current "best guess" of what the audience's experience would be."

Heckel (1991) characterizes software design as concerned primarily with communication, and we can say this even more convincingly about educational software. Heckel chooses filmmaking as an example because it "illustrates the transition from an engineering discipline to an art form".

Ellis (1982) compares the production of film and TV. Film production is described as a craft production, TV production as an industrial production. Cinema's typical product is the single film rather than a series, a prototype rather than a group of similar products. Cinema has often provided TV directly with prototypes for a series. Ellis (p. 189) states:
"Film production has become a form of craft production, producing prototypes with little continuity of personnel or experience from production to production. The characteristic mode of film production has a small group of entrepreneurs, often director alone or director with producer, aiming to assemble a "package". This package consists of an idea worked into a script (classically a dialogue script), together with commitments from key personnel such as stars, camera persons, editors, composers. This package is then touted around to various possible sources of production funding, always including distributors. If a distribution guarantee and enough money is raised, then the production goes ahead."

The tasks mentioned above are done by a fixed management team for TV, for a film by "free

1. Dave Hand, a Disney animation director, Heckel (1991, p. 179)
2. The story sketch should show characters, attitude, feelings, entertainment, expression, type of action, as well as telling the story of what is happening.

lance" managers/entrepreneurs. Another difference between film and TV production is the role of the director. With TV productions the director is working away from the floor where the actors are supervised by the "floor manager". The director works in an isolated room surrounded by technical equipment and staff. The director of a film is always present on set with the actors. In cinema (not in TV) production and distribution are separated completely, film production having more financial risks (Ellis, 1982). It is plausible to apply either the film production or the TV production metaphor to educational software development.

Finally, we can conclude that the experience of film making and TV making seems to be a good model for organizing and finding criteria to improve developing educational software. Two differences between film and TV making play an important role in this context: with film making production and distribution are separated completely and with film making for each project a unique team is formed. As this is not the case for regular TV we can say that film making is the better model for developing educational software. At least for the current types of educational software products that each have a unique character. The model of TV will become more important when educational software begins to be produced serially. Heines (1984) proposed TV as a good model for developing tutorial style educational software at the time. Compared with modern educational software products his products offer little interaction.

4.7 Questions for evaluation of the theory

The new theory can be summarized as a filling in of the aspects of the framework of Seligmann Wijers and Sol as follows.

• *First of all an explicit education or skill problem should exist. An organisation or person should exist who is able to behave as the owner of a problem (T1).*

• *The problem owner and the leader of the educational software development project should focus explicitly on the aspects education, information technology and media use (T2).*

• *To model the problem and possible solutions macro, meso, and micro levels should be distinguished explicitly (T3).*

• *Specifying the user interface should be carried out professionally (T4).*

• *Commitment of all persons involved should be based on clear definition and assignment of responsibilities. The required skills and disciplines should be represented within the team and the necessary communication within the team should be anticipated. Designing and realization should be carried out by separated teams (T5).*

• *The project leader should be able to behave as a director having a clear image of the final product and having senior experience to overview all disciplines involved in the development team (T6).*

• *Professional quality assurance measures are necessary (T7).*

• *Tools should support the transference of thought, passing the baton, within the team (T8).*

We will now derive 8 questions which will be used as criteria for evaluating our theory in the case studies, table 4.2. These questions describe significant aspects which may be found to be present to some degree during the development processes under study. In each case study the presence and influence of all questions will be investigated.

The questions are derived from the analyses of the different "ways of" in the previous chapters and of the discussion of the results in sections 4.1-4.5. The way of supporting is not essential for any method or project discussed in this study so far. Many very different automated tools and diagram techniques are being used, most for the realization phases of technical design and programming. Only in some research projects on educational software design are more advanced and experimental tools for the early design phases being used. Not the tools, but the way they are used may support the transference of thought. As it is difficult to evaluate the *use* of very different tools we do not formulate questions on supporting tools.

Aspect of the framework of Seligmann, Wijers and Sol	Questions
way of thinking (T1) and (T2)	1. How is the emphasis on instructional versus information technology aspects balanced? 2. Is the product under development considered as a tool and/or as a medium?
way of modelling (T3) and (T4)	3. Are the macro, meso and micro levels of modelling distinguished explicitly?
way of working (T5)	4. To what degree are different responsibilities within the development team defined and implemented? 5. To what degree is communication realized explicitly between different disciplines? 6. To what degree are the design and realization of the product separated into teams with different skills?
way of controlling (T6) and (T7)	7. Is the project directed by one person who has a clear image in mind of the characteristics of the final product? 8. To what degree are explicit quality assurance measures provided?
way of supporting (T8)	no questions

Table 4.2: Questions for evaluating the theory using case studies.

Question 1: How is the emphasis on instructional versus information technology aspects divided?
During the complete life cycle of development all necessary disciplines and skills must be involved at a professional level. This means that the disciplines, subject matter expertise, project management, instructional technology, media use and software engineering are needed in a balanced way. If one of these disciplines is neglected or not available at a professional level the risk of failure is increased considerably.

Question 2: Is the product under development considered as a tool and/or as a medium?
The tool perspective is very common in information system development. It is based on the view that people use tools to perform tasks. Information flows from the user through the user

interface into the application (semantics of the tool). To achieve the aims of educational software there has to be more than getting something learned in the real world. Educational software has to be considered as a medium rather than as a tool. This demands integration of several disciplines including the discipline of "communication". The environment in which learning or training takes place has to elicit direct engagement. WHY and WHAT is offered (presented) to the student is important, HOW it is offered (presented), and HOW interaction is organized are equally important.

Question 3: Are the macro, meso, and micro levels of modelling distinguished explicitly?
The classes of design problems distinguished at the macro, meso, and micro levels must be recognized during design and realization. Development is too complex to base it on an holistic approach in which problems at all levels are tackled simultaneously. These steps can be described as organisational, instructional and interaction oriented, each to be associated with a different discipline.

Question 4: To what degree are different responsibilities within the development team defined and implemented?
As many different disciplines are involved in educational software development it is necessary to describe main tasks, functions and required skills explicitly before a project starts. It is not selfevident who has to be committed to what task. Usually it is not easy to recognize the different tasks that have to be carried out iteratively during the macro, meso and micro levels of design. Definition of subject matter components, learning objectives, didactic strategies, interaction styles and media effects require for example different expertise and responsibility. The members of the team have to know their own responsibility and that of their colleagues. Having a skill does not imply having responsibility for all tasks requiring that skill. When one person owns more skills from different disciplines, e.g. subject matter expertise and media expertise, then the final responsibility for each of the expertise within the team must be agreed clearly or even "contracted".

Question 5: To what degree is communication realized explicitly between different disciplines?
Communication within a team between the members with an engineering background and those with a background in other sciences and domains is usually very difficult. Nevertheless, transference of didactic thought is the critical process in educational software development. As a consequence of this, communication between disciplines within a team should be organized and supported explicitly. The most difficult collaborations and communications are those between subject matter expert and educational technologist and between educational technologist and software engineer. Communication within a team may improve when experience in more than one discipline is embodied in one person.

Question 6: To what degree are the design and realization of the product separated between teams with different skills?
Design and realization need different skills from different disciplines. In professional projects design and realization should be carried out by two different teams. Communication between both teams must be monitored by the director of the whole project.

Question 7: Is the project directed by one person who has a clear image in mind of the characteristics of the final product?
From the film making metaphor and from the Teleac project (section 3.2) it can be derived that

the role of the director can not be overestimated. She supervises team members from all the disciplines involved in the project. The director should have great commitment to and final responsibility for quality control.

Question 8: To what degree are explicit quality assurance measures provided?
Developing a piece of educational software must be considered to be a unique project every time. Context and conditions are always different and unique. A team must be formed using a professional project management method suited to the medium of educational software for each new project. The method should support both educational design and technical design and realization. Funding of the project should be planned and achieved using professional techniques. A quality assurance plan should be made at the start of a project. This must be updated after each milestone.

These eight questions will be answered in three case studies described in the next chapters. The criteria for selection of the cases were:

- the team-approach was used;
- the subject matter had rather different topics interesting for many students (about 1000 or more);
- the product was finished recently;
- the product provided about 10 hours of educational software or more;
- the first experiences with use were available;
- the project documents and the developers were available for analysis and interviews.

The first case is an in-house project for training in banking. The second is an out-house project for training in aircraft maintenance. The third is a multi-partner project for training Dutch as a second language. By choosing an in-house, an out-house and a multi-partner project it was expected that the theory would be tested using a different spectrum of possible project conditions. The answers found are compared and used to answer the general research question of this study in chapter 8.

5. CASE 1: TRAINING BANK EMPLOYEES

This case study was carried out within banking. Educational software in banking has been used for more than 10 years on a large scale, with many students per educational software product (Kunneman and Hertgers, 1992). Banks are professional organizations demanding a high quality for the products they sell and use, including training and education. This case was selected for several reasons. First, educational software is not that new in banking, so experience with developing and using can be expected in the sector. Second, this was an *in-house* project with internal project management, with only a minor number of assignments for external experts. Third, the product was accepted and finished for use at a large scale, 1000 or more students. Fourth, use of the product is not compulsory; local banks are free to buy and use the product. Fifth, the educational software produced provides more than 10 hours of session time and it is integrated with other media.

The context of the project is described in general in 5.1. The product is described in 5.2. All available project documents were studied and the project leader and some team members were interviewed. In sections 5.3-5.6 the development process is analyzed using the framework presented in chapter 2. Finally, the research questions are answered in 5.7.

5.1 Rabobank Nederland

The first co-operative bank in the Netherlands was founded in 1895. Within a period of twenty years hundreds of such local banks existed named Raiffeisen Bank or Boerenleenbank. In 1898 two central banks were founded to support all independent local member banks with each keeping their own rights and responsibilities. In 1972 these two central banks merged completely to form the Rabobank Nederland. In 1993 the about 750 co-operative Rabobanks had more than 36,000 employees including Rabobank Nederland. Rabobank Nederland has also more than 50 offices worldwide.

The aim of Rabobank Nederland is to support the participating local banks. An example of a supporting function is Education and Training and the design and implementation of training programs for (new) employees of the local banks selling new products and services.

The organization of Central Rabobank Nederland is depicted in figure 5.1. Education and Training[1] (ET) is a central department for all local banks with the following goals:
- to develop a policy for the education and training needed in the market;
- to develop and maintain courses;
- to evaluate quality, effectiveness and efficiency of courses and training programs;
- to co-ordinate scheduled course programs;
- to organize courses, seminars and training by third parties.

The following disciplines are represented in the department Education and Training: educational technology, subject skills on bank affairs, commercial skills and project management. During the project described here the department collaborated with the Computer Based Training (CBT) group (see fig. 5.1). The CBT-group consisted of a small number of CBT designers and programmers.

1. Officially, the name of the department is Corporate Sector Education and Training.

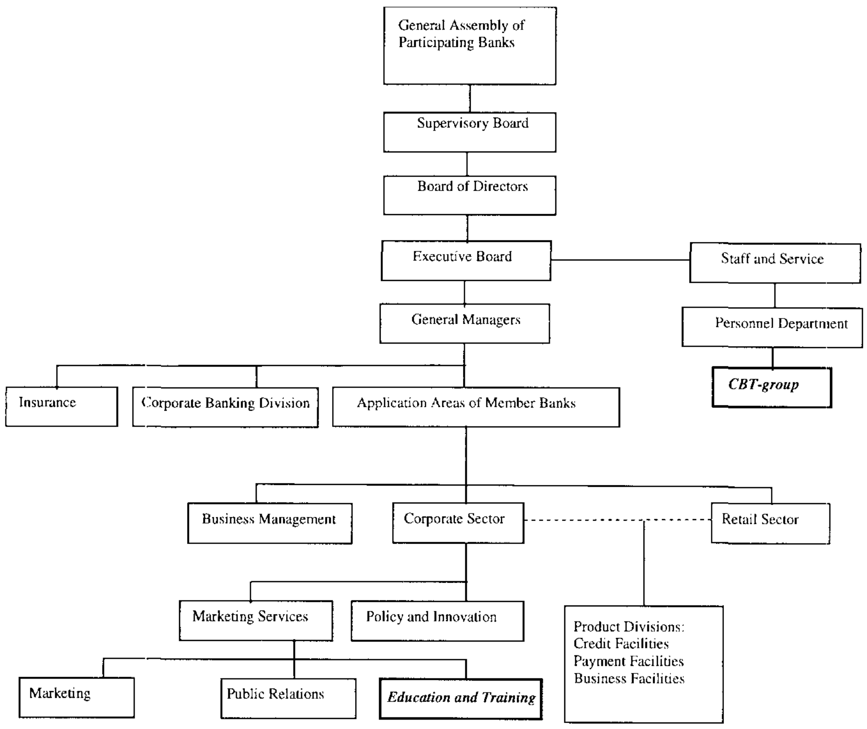


Figure 5.1 Organization of Rabobank Nederland showing the position of the department of *Education and Training* and the *CBT-group*.

5.2 The *BZD* course

The local banks have wanted a comprehensive, basic course, for counter employees on assisting clients from the business community and institutions for many years. This demand resulted in the development of a multimedia educational software package teaching the course Basisopleiding Zakelijke Dienstverlening[1] (*BZD*). The target group for the course are employees at the business counters, in Dutch Baliemedewerkers Bedrijven[2], BMB. The main reasons for this request were:

1. the way a BMB functions was not satisfactory and did not fulfill internal functional requirements, or the requirements of the clients, or did not conform to modern concepts of information technology provision by the bank;
2. the current BMB worker was not well qualified for the changing requirements of the near future;
3. the average BMB worker was not very flexible, this is caused by a low level of education and lack of breadth of experience;
4. employees from other functional counters who sometimes assist business clients are

1. In English: Basic Education Business Services.
2. In English: Counter Employees Business Services.

inadequately trained in the service concepts required for business clients.

The current requirements of the function BMB worker were described in general terms without criteria for quality. In most cases the knowledge required, skills, attitude and level of mastery were not described in enough detail. The management of the local banks, in consultation with Rabobank Nederland, considered several different ways to improve the BMB worker's flexibility; among others, re-engineering the organization, changing the responsibilities of the employees or replacing employees by better qualified people. Finally it was decided to offer training to current and new BMB workers. As educational software has become a proven means for training in Banks (Kunneman and Hertgers, 1992) it was proposed to develop a multimedia course of which educational software would be a component.

The department Education and Training initiated a proposal for a project to realize a comprehensive course for all functions at the business counter. The Corporate Sector director was made responsible principle for the project, to work out the description of the project goals and to form a steering committee with representatives from all product divisions (Payments, Business Financing, Credits, Foreign Affairs and Insurances). This resulted in a project to develop a course with the following modules: a general introductory module, a module for each subject area (product group) and the related administration, commercial and marketing aspects, a module for commercial skills training and a second optional module for financing. It was required that the course should focus on the tasks a BMB worker carries out.

5.2.1 GENERAL PROJECT REQUIREMENTS

The following requirements for the contents of the course were specified in the project assignment:

1. the constraints and the contents of the modules should be based on actual function profiles[1];
2. the required pre-knowledge for the students should be described in terms of education and experience;
3. operational objectives should be described as controllable operations and measurable results.

The following requirements for the instructional strategies were specified:

1. the course should be flexible because requirements at local bank level for scheduling and individual needs differ. This flexibility should be implemented by:
- modular partitioning of the contents;
- tuning between the modules;
- standard structure of the content of each module;
- consistency in didactic styles;
- adaptation to former education and continuation courses;
2. it should be possible to follow the course locally with available resources and staff;
3. the time required to follow the course should be minimal.

With respect to evaluation the following requirements were given:

1. an instrument should be developed to measure and compare results after completing the modules;
2. scoring of the training process and the training result should be realized by formative and summative evaluation[2];
3. the evaluation results should be suitable for future revisions of the course.

1. A function profile is defined in section 5.4.

The formal project assignment described four elements to be developed:
1. an introductory module aiming that a BMB worker that would be able to work independently according to their function profile within a period of 6 months, spending one day a week on the course. This module contains subject matter, administration and commercial aspects;
2. separate modules for special product-market combinations (payments, financing, credits, foreign money and insurances);
3. commercial training using standard media but with integrated contents;
4. optional modules for special product-market combinations (this point later fell).

The following products should be delivered for each module: an instructors guide, a student guide, learning resources: text books, work books, exercise materials and videotapes; training resources: work books, cases, role-games; an examination instrument; an evaluation instrument; a distance-consultancy service.

The project team was organized as depicted in table 5.1. The project team consisted of all persons mentioned in the column Function. The subprojects for developing the modules consisted of the project leader and one or more subject matter experts of the product groups (right column). The project team was augmented by some extra educational and media experts from other departments of Rabobank Nederland. When needed third-parties were hired to do special tasks. The general module was developed by the department of Education and Training assisted by specialists from other disciplines e.g. automation, security and legal affairs. The final product is presented before the development process is described and analyzed.

Function	From the group
Project leader	Marketing Services / Education and Training
Domestic payments subject matter expert	Product Division Payment Facilities
Credits subject matter expert	Product Division Credit Facilities
Business financing subject matter expert	Product Division Business Financing
Insurances subject matter expert	Product Division Insurances
Foreign payments subject matter expert	Functional Division Foreign Affairs
Commercial expert	Marketing Services

Table 5.1: Organisation of the *BZD* team

5.2.2 THE COURSE

Internal dissemination of the course was started in summer 1992. The following products were available:
- a manual for local instructors;
- a student guide;
- an introduction module (1) consisting of a videotape (V), a textbook (T) and educational software (E);
- introductory commercial training of two days (2);

2. Formative evaluation is carried out (with a few users from the target group) to evaluate and improve the *design*. Summative evaluation is carried out with large numbers of users during realistic *use* of the product, see 1.2.

- a module (3) domestic payments (V, T, E);
- a module (4) foreign payments (V, T, E);
- a module (5) credits (V, T, E);
- a computer based test on the subject matter of the modules mentioned above (E);
- commercial training of three days (6);
- a module (7) financing I (T, E);
- a module (8) insurances (T, E);
- a computer based examination on all the subject matter of the course (E);
- an optional module (9) financing II (T, E);

about 1 month	Introduction (1)	V, T, E
	Commercial Training (2)	traditional
about 3 month	Domestic Payments (3)	V, T, E
	Foreign Payments (4)	V, T, E
	Credits (5)	V, T, E
	Computer based Test	E
	Commercial Training (6)	traditional
about 2 month	Financing I (7)	T, E
	Insurances (8)	T, E
	Examination	E
about 2 month (optional)	Financing II (9)	T, E
	Examination	E

Figure 5.2 Structure of the multimedia course for *BZD* students showing the planned duration of each block of modules, indicated in months, based on part time training. Media: video (V), textbook (T) and educational software (E)

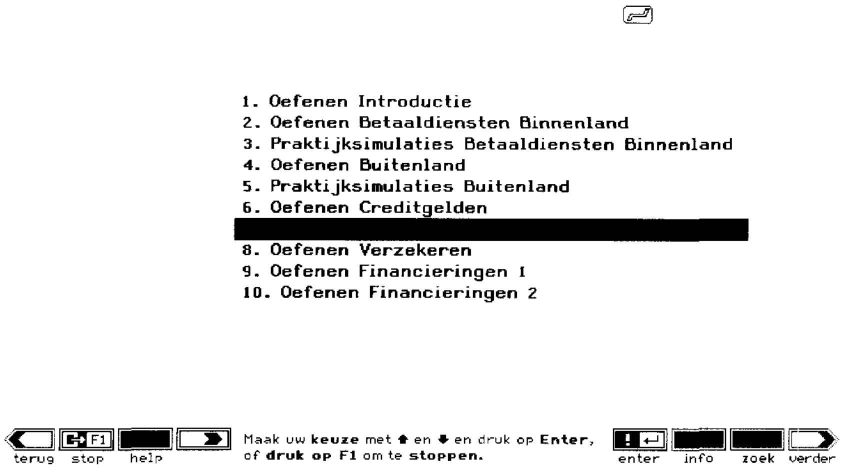
These modules had to be studied part-time (1 day a week) for about 6 month, see figure 5.2. The estimated total time for study is 6 (month) x 4 (week) x 8 (hour) = 192[1] hours. The optional module needs another 2 months. The standard order for following the modules of the courses is the one given above. Not all modules have a video component. The video is presented first (30 minutes). When the textbook has been studied by the student (10-15 hours) (s)he can use computer based exercises and tests (4 hours). After this the student must follow a CBT simulation for most modules (1 hour).

A license to use one or more modules can be purchased by a local bank with a subscription for revisions. With each license one student at a time can follow the modules, reduced prices are offered for larger numbers of students. During the first year about 680 complete courses were sold. This *BZD* course can be used in combination with general NIBE-courses[2] already available on the educational market for banks in the Netherlands. Two examples of screens are presented in figure 5.3 and 5.4. One year after first publication the contents of all educational software components were updated.

1. The total amount of educational software is about 17 hours of session time.
2. NIBE: Nederlands Instituut voor Bank- en Effectenbedrijf (Netherlands Institute for Banking and Stock-jobbing). NIBE is managed by the Dutch banks and provides general courses concerning some branches and products of banking in the Netherlands

The instructional structure of the non-commercial modules (1,3,4,5,6,7 and 8) of the course consists of the same steps and components.

The *BZD* product in total consists of about 7 hours of exercises, 7 hours of simulations and 3 hours of tests. 2 hours of video and textbooks for all modules were also produced[1].

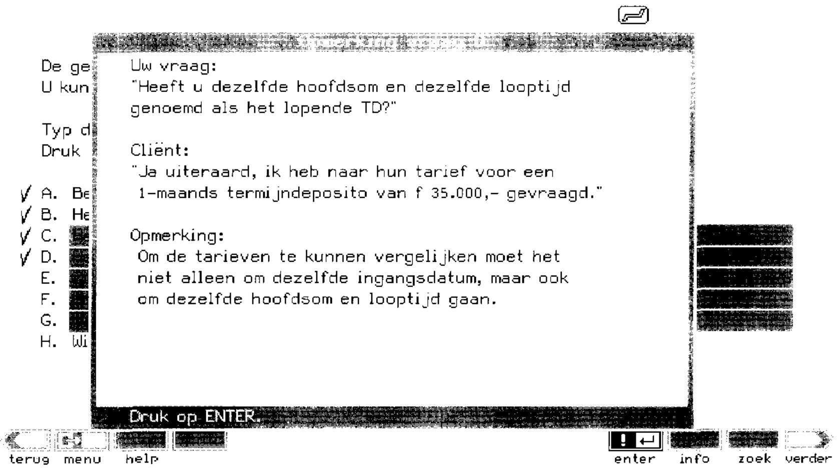


1. Oefenen Introductie
2. Oefenen Betaaldiensten Binnenland
3. Praktijksimulaties Betaaldiensten Binnenland
4. Oefenen Buitenland
5. Praktijksimulaties Buitenland
6. Oefenen Creditgelden

8. Oefenen Verzekeren
9. Oefenen Financieringen 1
10. Oefenen Financieringen 2

Maak uw **keuze** met ↑ en ↓ en druk op **Enter**, of **druk op F1** om te **stoppen.**

Figure 5.3 Main menu of the course pointing to the choice *7. Simulation of Credits.* Every choice represents one hour session time or more. For choice 7 five complete simulations are provided.



De ge
U kun

Typ d
Druk

✓ A. Be
✓ B. He
✓ C.
✓ D.
E.
F.
G.
H. Wi

Uw vraag:
"Heeft u dezelfde hoofdsom en dezelfde looptijd genoemd als het lopende TD?"

Cliënt:
"Ja uiteraard, ik heb naar hun tarief voor een 1-maands termijndeposito van f 35.000,- gevraagd."

Opmerking:
Om de tarieven te kunnen vergelijken moet het niet alleen om dezelfde ingangsdatum, maar ook om dezelfde hoofdsom en looptijd gaan.

Druk op ENTER.

Figure 5.4 A screen presenting feedback to the student on their dialogue with the client during the "Competitor" case of the module 7. Simulation of Credits. See tables 5.8-5.10 for more details.

1. According to the project leader about 1.6 man year of Rabobank employees were spent to produce all products mentioned here.

5.3 Way of thinking

The project was initiated to try to solve the educational problems of BMB employees. The product aimed at was considered to be a strategic product for improvement of one of the primary functions of the bank. The strategic importance was emphasized by a quantitative analysis of the internal market for such a product. Questionnaires were sent to 150 local banks. Much quantitative and qualitative data were collected. The final result was a prognosis of the need for education, but not for the use of the course.

An important characteristic of this project is the ambivalent situation that the target group of students within the Rabobank is well known and can be analyzed well but that the use of the course is not guaranteed because of the existence of an internal market mechanism for using the educational software package. It is a closed market of the business counter employees (and their local managers) of all local Rabobanks. The existence of a closed market is a special condition of this project, commitment to use the course had to be won by promoting the product. Although the potential users were known there was no way to involve them in the definition of acceptance criteria and procedures. These had to be defined "on behalf of" the users by some selection of users. This introduced extra risks for the fitting of the product to the needs of the users and the future use of the course.

A second characteristic is that all subject matter knowledge and also the other expertise needed to perform such a project was available in-house, albeit dispersed in different departments of Rabobank Nederland. This included expertise on instructional design, educational software technology, video production, technical writing and project management. This expertise was, however, found difficult to manage by the project leader at Education and Training. It was not obvious beforehand how to arrange for the total commitment of experienced persons from different departments. Although the project leader did not have the power to demand commitment from all team members and from supporting departments directly, not even via the steering committee, he had the budget to hire external expertise from third-parties to do relatively small assignments, e.g. performing media selection and writing video scripts for the modules.

A third characteristic is that the project team as a team had no real experience with developing educational software. The project leader and most team members had not participated in earlier projects where analysis, design and realization of multimedia materials was covered, resulting in more disciplines in the team. The project leader decided at an early stage that strong project management would be necessary. Only experience with general professional project management was available in the department Education and Training. A general method for project management (Wijnen et al., 1988) was chosen. This method had to be adapted to educational design and for software development aspects[1]. Only modest experience in developing educational software was present in the team, experience with recognizing the advantages of educational software was also lacking on the part of the team. The subject matter experts especially, senior banking experts, did not have experience with the design process or with information technology. The conclusion is that this project, although a strategic one, also had strong learning aspects in it. The subject matter experts e.g. had to be instructed explicitly on the possibilities of educational software before some parts of the design process could start.

The project was supervised by a department of Education and Training with extensive

1. The general method described in Wijnen et al. (1988) is not adequate for systems development. The method was extended on an ad hoc basis.

experience in instructional design. The team was very skilled in curriculum analysis, task analysis and instructional design. Education and Training is a subdepartment of Marketing Services, which has a great influence on the approach applied to instructional designing. Based on the large experience with instructional design within the department it was stated that: "We are convinced that many courses do not effect functioning at the work place itself or have too long development times because the discussions on the requirements are continuously repeated during the development process. A good definition of the function profiles involved and a good task analysis during the definition phase may avoid these problems." This new training course was also considered as a introduction to the existing traditional commercial courses offered to employees and for this reason the concepts of commercial behavior and communication had to be exactly the same as in the commercial training program.

From the start of the project on, the project team of Education and Training knew that a strong emphasis on *general* project management was required to co-ordinate and control communication between the different disciplines in the team (way of working). This resulted also in emphasis on subject matter analysis and educational design. In the initial phase of the project prototyping and try-outs were carried out. This means that to some degree user participation was realized for the purpose of formative evaluation. Comprehensive user participation to prepare the user organization for implementing the course was impossible because of the market concept behind the project; good promotion was substituted for this.

The last characteristic to be mentioned here is the fact that the project leader, as an educational technologist, did not have a subject matter background. Thus, it was not easy for him to evaluate and test the designs and the products in each stage of development personally. Nevertheless, he was the central co-ordinator for all resources of the project.

The way of thinking can be summarized as follows:
- the product aimed at is considered to be strategic;
- the local banks are free to buy the product; so use is not guaranteed and promotion should be provided;
- most internal subject matter and other experts are working in different departments, their commitment had to be arranged explicitly;
- within the design team strong expertise was available on instructional design;
- little experience was available with systems development and software engineering;
- the project leader had no expertise on the subject matter. He was a general educational technologist and an experienced project manager. Developing educational software was a new job for him.

5.4 Way of modelling

As described in the previous section the discipline of instructional design was dominant in the project. Therefore the way of modelling was based on the tradition of instructional design. The educational analysis and design were structured strongly by modelling concepts (fig 5.5[1]).

An extra problem for this type of training of procedural tasks is that the definition of the subject matter to be taught/learnt is not straight forward, nor are the content of the products and services to sell at the counter.

The operations the employees have to perform are described in great detail. Therefore, modelling the situation is extremely important. To develop vocational courses it is necessary

1. These macro, meso, micro levels are defined within this project. They are not the definition given in chapter 4.

to specify previously an explicit function profile and to perform an accurate task analysis. This should increase the efficiency of the further development process and it should make the course more effective. Function profiles must be the basis for deriving the learning objectives of the course.

function analysis

task profiles

filling in Task Analysis Matrices
deriving learning objectives

subject matter description for the textbook

designing all other media per module

(CBT-simulations, role-games, video's, tests)

Figure 5.5 Subsequent modelling concepts during design

The first step in the modeling process is to describe the qualifications for all possible functions of current and future BMB workers using function profiles based on the "spinning top-model"[1] (Kwantes et al., 1991). A function profile is defined as "*a set of criteria required for good functioning of an employee in a well-defined class of related functions*". A task profile is a specialization of the function profile: "*A task profile must support the identification of current and future task elements of an individual employee with a specific function*". Three types of tasks are defined for the BMB employee concerning:

- the characteristics of contents and form of a product of the bank (bank technical subject matters, e.g. payments, credits, insurance);
- selling oriented behavior of the employee (commercial tasks);
- administrative and logistic acting (administrative tasks).

Task	Objectives
task area: credits	increasing the position in the market for long-term credits by 25% in this year
main task: selling a specific credit product	increasing the sale of standard products by 15% more long-term money
partial task: sale presentation "koopsompolis"	individually contacting potential clients to sell "koopsompolis"
task element: presenting characteristics of the product "koopsompolis" effectively	presenting quality of the product and individual advantages to the client
action: speaking	speaking clearly and using an enthusiastic tone

Table 5.2: Example of relation between tasks and objectives (from Kwantes et al., 1991).

Different levels of behavior by the BMB employee can be identified for each type of task. Using this model "reference functions" can be defined based on tasks and levels. Tasks are analyzed down to the most elementary level of task components and acts. This analysis results

1. In Dutch "tol"-model. A tol (top) is a spinning toy with a cone form. Thijssen (1988) presented this conceptual model to distinguish between hierarchical "levels" of tasks.

in a task-analysis-matrix TAM (see Kwantes et al., 1991). To analyse a task in relation to the objective it is contributing to, a systematic approach is necessary. In table 5.2 some examples are given of the relationship between tasks and objectives at different levels.

5.4.1 THE TASK-ANALYSIS-MATRIX

All functions of the BMB employees were specified with many TAM's. The products and services the employee has to cover are mentioned in the rows of the TAM, in the columns all actions necessary to "deliver" a complete product to the client at the counter. Each cell of a TAM represents a possible task that is marked if it is essential for the function. Each of the three types of tasks mentioned above is divided into "roles" (levels of behaviour). For commercial tasks these are e.g.:

- referring role (A): the employee does not explain a product to the client but refers the client to a colleague;
- informing role (B): the employee explains the product but refers the client to a colleague for the final offer to the client. This role is based on the communication model: specify situation, analyse the problem, offer a solution, conclude and carry out[1];
- advising role (C): the employee is able to explain all ins and outs of the product and to sell it.

An example of a TAM is given in table 5.3. In a TAM the tasks and both current and future function profiles can be filled in. The roles for the bank technical tasks are: start up role, informing role and maintenance/guarding role. The roles for the administrative tasks are: registering role, maintenance role, analyzing role and co-ordinating role.

	Referring role	Informing role	Advising role
payments		x	x
financing	x		
credits		x	+
foreign affairs	x	+	
insurances	x		

Table 5.3: Example of a TAM on commercial tasks for one specific function of a BMB employee at the highest level (x: current role; +: future role); rows: products.

TAM's were filled in for all relevant products and also with tasks differentiated to some general characteristic of local banks in cities and small villages. From these a set of "mean" TAM's were derived for the "standard" local bank. This set was used as the basis for the learning objectives and further design of the *BZD* course. These TAM's were constructed by senior staff from some local banks supervised by educational specialists from Education and Training. This task analysis was carried out at three levels of detail (macro[2], meso, micro). At the lowest level elementary tasks and actions of individual employees were identified. An example of the micro

1. In Dutch SPAAR-formula: Situatiebeschrijving, Probleemanalyse, Aanbieden, Afhandelen, Regelen. This formula is well known to the students from other courses given at Rabobank.
2. The macro, meso and micro are here defined by Rabobank within this project. They are different from the definition given in chapter 4.

level is given in table 5.4

	Required role	Start of talk	Analysis of goals	Referring	Problem inspection	Problem analysis	Giving information
domestic payments	advising	x				x	
foreign payments	referring	x	x	x			
credits	referring	x	x	x			
insur-ances	informing	x			x		x
financing	referring	x	x	x			

Table 5.4: An example of a TAM for one specific function of a BMB employee at the micro level (Kwantes et al., 1991). Only a subset of the columns is depicted here.

5.4.2 LEARNING OBJECTIVES

After the task analysis using TAM's the next problem was to identify learning objectives. Kwantes and Thijssen (1986) have developed a taxonomy to classify learning objectives. They distinguish two categories of knowledge, facts and concepts and two categories of skills, reproductive and productive. These are ordered hierarchically. They also distinguish two domains of knowledge: units and structures or schedules. Units are terms or objects with a single meaning, e.g. words, symbols. Structures or schedules contain complex knowledge, e.g. procedures, methods. Four skill domains are also distinguished: cognitive; motor, also body language and mimicry; affective and social. These domains are ordered in tables 5.5 and 5.6. Learning objectives were derived in three steps from the micro-level TAM's of each product. First the goals for the reproductive skills are described, then the goals for knowledge of facts and finally the knowledge of concepts. In this project not all categories were used to specify the goals of the *BZD* course.

	Terms/objects	Structures/schedules
facts	+	+
concepts	+	+

Table 5.5: Categories of knowledge (Kwantes and Thijssen, 1986). Legend: +: supported in non-commercial training; x: supported in commercial training.

	Cognitive	Motor	Affective	Social
reproductive	x	x	x	x
productive	--	--	--	--

Table 5.6: Categories of skills (Kwantes and Thijssen, 1986). Legend: +: supported in non-commercial training; x: supported in commercial training.

The same subject matter experts who filled in the TAM's were later asked to identify and to describe learning objectives. A special manual with instructions and examples was therefore produced for them by the educational specialists. The learning objectives were derived directly from TAM's by explaining in words what was identified in TAM's with marks. For this purpose special forms were developed to be filled in by the subject matter experts. The same forms containing specifications of the content were used later for producing the draft for the textbooks.

After instructing the subject matter experts, the learning objectives were completed in several iterations. The educational specialists of Education and Training then edited the texts describing the objectives. The subject matter experts reviewed and improved the texts again, all the drafts of all textbooks were made this way. These drafts were the "formal" definitions of the subject matter and all other learning materials, videos, cases, exercises, simulations and tests were derived from these drafts after the phase of media selection was completed.

5.4.3 EDUCATIONAL DESIGN AND MEDIA SELECTION

Starting from the initial definition of the main modules for the different product groups and the definition of the subject matter in the drafts of the textbooks, media selection and the educational design for each medium was carried out. The following learning steps were identified based on the theories of Gagné and Briggs (1974).

learning step 1:	preparation for learning, gain attention: inform the student about the behaviour to be learnt (objectives); inform the student about content and goal of each module;
learning step 2:	acquisition of knowledge: present subject matter to the student; provide opportunity to memorize;
learning step 3:	processing of knowledge: provide exercises on subject matter in advising/selling situations about all products presented so far;
learning step 4:	evaluation of learning results.

These learning phases were used for the educational design of the introduction module, the commercial training and the product modules. The following model was specified for the educational software of the product modules, see figure 5.6.

The learning steps of Gagné and Briggs are worked out for each module. Then what was the best medium for each step was decided: text, video, educational software or exercises/role-games. A numerical media selection method (Leiblum, 1980) was used for this purpose. This selection method can be considered to be structured decision making. The first column contains a list of 34 "learning attribute mechanisms" prescribed by Leiblum and divided into five groups:

1. instructional event or task requirements;
2. learner requirements;
3. sensory modality requirements;
4. administrative requirements;
5. distribution or local environment requirements.

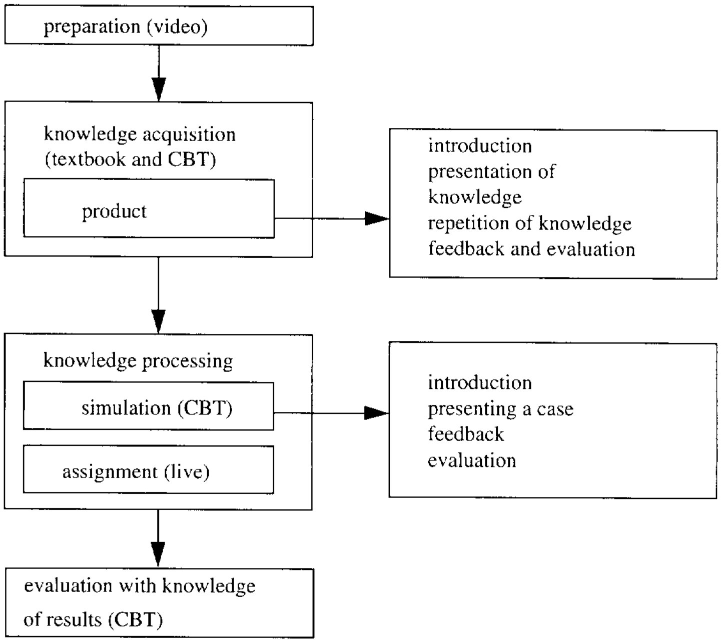


Figure 5.6 Educational design for the product modules using the learning phases of Gagné. The results of the media selection procedure are mentioned in parenthesis.

distribution or local environment requirements	importance factor	Medium 1 text	Medium 2 video	Medium 3 CBT
1. ability to adapt to individual needs	10 *	1 = 10	0 = 0	3 = 30
5. ability to produce life like images	10 *	2 = 20	3 = 30	3 = 30
16. ability to provide learner control	5 *	3 = 15	1 = 5	3 = 15
21. motion	10 *	0 = 0	3 = 30	1 = 10
22. easy accessibility	10 *	3 = 30	1 = 10	1 = 10
29. scheduling flexibility	10 *	3 = 30	1 = 10	1 = 10
TOTALS (over rows 1-34)		320	253	427

Table 5.7: Example of the numerical selection method (Leiblum, 1980) for learning step 2, knowledge presentation and one element of subject matter, domestic payments. A few rows (1,5,16..) only are presented as examples.

The developer then determines how important the different learning attribute mechanisms are in the context of the content, the target group and the didactical goals. This is called the importance factor (0 = no importance; 10 = very important) and written in column two. The next three (or more) columns are used to rate the relevant media, in this case text, video and educational software. The media's ability to perform the task mentioned on each row is rated (0 = cannot fulfil the task; 3 = can fulfil the task very well). The importance factor and the rate are multiplied for each cell to give a medium score; the total gives a quantitative indication of its appropriateness for each medium, see table 5.7.

5.4.4 SIMULATIONS

One standard model for all simulations of the dialogue between student and client at the business counter of the bank was designed. Again, observing and interviewing senior BMB employees was the basis of this design.

It is important to know that during design of the case simulations emphasis was put on the specification of one model for all simulations of the cases. Verification and evaluation of this model was achieved using user participation. Once the definitive model was approved it formed the basis for further design. In this way all the cases were a slight variation of the model. This gave a consistent view to the students. Many dialogs were designed and prototyped to refine and evaluate the standard model. Co-operation between designers and subject matter experts was quite heavy during this phase because the subject matter experts did not have experience in and feeling for simulation design. Elicitation of their experience was done by personal contacts and interviews. The final general design is depicted in figure 5.7. This standard model was also used to structure all videos for the product modules, so the students could recognize, during the simulations, situations already seen during the introductory video. The cases for commercial training were structured using the same model.

start
describe the case problem
ask for preliminary solution
simulate a diagnostic dialog with client
discuss/evaluate the dialog
invite the student to propose a solution to the client
discuss/evaluate chosen solution
discuss student's role in communication
invite the student to give extra services/advise to the client
discuss/evaluate chosen extra services/advise
summarize and evaluate the simulation
stop

Figure 5.7 The standard model for all the simulations of clients at the counter presented to the student during the cases (see tables 5.8-5.10).

In tables 5.8, 5.9 and 5.10 an example of a simulation is given as a summary of the possible elements of the dialogue. The subject of the simulation "the competitor" is the difference of the interest between different competing banks. All texts are presented in windows to the student, see figure 5.4. The problem description is given in table 5.8. The student has to ask the client at the counter, one or more questions to collect information to solve a problem. The questions must be chosen by multiple choice. After each choice the simulated answer of the client is presented to the student in another window on the screen. The student will receive comments on the relevance of the question s/he asks. The student can decide to stop further questioning. The student then has to give information to the client, see table 5.10 for some alternative texts.

S/he can again ask for comments on the informations he gave to the client. This type of simulation is based on multiple choice from small written texts.

"A client calls you and they are very irritated.
Yesterday they got the account-list of the extended deposit.
Now, he discovers that the Holland Bank offers 0.25% more interest for his money."

Table 5.8: Problem description of the simulation on Credits presented to the student on the screen (case: the Competitor)

Question	Answer	Comment
Do you know that the difference of the interest is only a few guilders?	Yes, but that is also a few for the bank!	Probably this is right, but is no argument for the client. This question increases the irritation of the client.
Do you also have an account at the Holland Bank?	No.	This question does not have any sense.
Did you contact the Holland Bank or did the Holland Bank contact you?	They called me.	This answer shows that the Holland Bank is active by recruiting. It is clear that attractive offers are being made to the client.
Did you mention the same total amount and the same currency as your current deposit?	Yes of course. I asked for their rate for a 1 month deposit of DFL 35,000.	To compare the rates both the total amount and the starting date must be the same.
Have the rates for deposit changed in the last days?	I've not got the faintest idea!	This question is very relevant, but you should not ask this question to the client. You should consult internally. For example it could have been possible that the rates increased by 0.2% yesterday. In this case the current rate of Rabobank would be the same as the rate of Holland Bank.
How do you know that the Holland Bank is offering a higher interest rate?	They called me today.	It is important to know the reliability of the source of the client. If the rate is from hearsay then you may doubt the reliability.
When did you contact the Holland Bank?	This morning.	Our rate and Holland Bank's rate are from different dates. In between the interest and so the rate for deposits may have been changed.
Would you like to take out your deposit prematurely?	Yes, if that is possible!	A deposit cannot be taken out prematurely. So, you should not suggest this.

Table 5.9: Diagnostic dialogue simulation; case: the "Competitor"; the 4th row presents the feedback displayed in figure 5.4. *(italics indicates right question)*

Information	Comments to student
The competition with respect to deposit is very heavy.	Yes, that's true, but with this information you urge the client to ask for information from competitors.
The interest on the financial market can change daily.	The premature conclusion of the client was mainly caused by the fact that he did not realize that interest can change daily.
The level of todays rate for deposits from the local Rabobank.	Only the current interest rate of the local Rabobank can be compared with the current interest rate of the competitor. The interest rate of both may change daily.
The interest rates for deposit are derived directly from the financial market.	That's the reason this interest may change daily.
It is generally auspicious to keep deposits on the current account.	This is irrelevant for the client's problem.

Table 5.10: What information can you give to the client? *(italics means right information)*

5.5 Way of working

The chosen way of working was already described in the project assignment (result of phase 0). This is an adaptation (see 5.3) of a general method for project management (Wijnen et al., 1988). This method was designed for general purposes project management e.g. development of objects (houses, other constructions), product development, knowledge development (e.g. looking for new markets) and development of organizations. The original method[1] can be summarized as follows:

Phase 0: Initiative. Product: project assignment and quality assurance plan;
Phase 1: Definition. Product: program of requirements;
Phase 2: Design. Product: functional design of solution;
Phase 3: Preparation. Product: technical design for realization;
Phase 4: Realization. Product: working solution;
Phase 5: Maintenance. Product: working improved solution.

The method was adapted for this project in the following way. The following is an abbreviated list presented during phase 0 of the project assignment.

Phase 0:	Initiative.
	Problem analysis
	Accepted project analysis and quality assurance plan.
Phase 1:	Definition.
	Analysis of functions
	Task analysis
	Analysis of target group
	Analysis of needs for education and training
	Specification of general goals of the project
	Description of project requirements
	Project planning
	Refining the quality assurance plan

1. This method was not applied to the last phases (3,4 and 5) of the *BZD* project.

Phase 2:	Design.
	Learning objectives
	Media selection
	Educational design
	Detailed subject matter definition (draft for text books)
	Evaluation standards
	Review of the design
	Prototyping and try outs
Phase 3:	Preparation.
	Technical design of all components
	Refining planning
	Contracting third parties
Phase 4:	Realization.
	Producing the product following technical design
	Acceptance procedure
	Preparation of maintenance
Phase 5:	Maintenance.
	Implementation in the organisation
	Final evaluation of the product and project
	Transfer of the maintenance plan
	Revision of the product

This way of working is composed mainly of sequential phases. The project was separated into subprojects for each module. After the definition phase of each module independent teams were formed to work simultaneously if needed. The co-ordination of all subprojects was done by the project leader from ET. The final responsibility for the contents of each module remained with the subject matter experts for that module. The design of the video and educational software products was done during the realization phase, not separated explicitly into design and production phases.

A formal report was produced only for the phases 0 and 1 according to the quality assurance plan. The project management style changed from a formal one during phases 0 and 1 into an informal style during the next phases.

5.6 Way of controlling

The method of project management chosen for this project (Wijnen et al., 1988) provides five aspects of control: time, budgets, quality, information (documentation) and organization. Following these aspects, after phase 1 (definition) an external audit was made of the quality of the project management and the results so far. The general conclusion was at the time, that according to the interviews given by the teams the quality of the results (effectiveness) was sufficient but that efficiency could be improved. The fact that the teams were inexperienced was given as the main reason for the low efficiency. For the next phases it was advised that more emphasis should be placed on quality assurance methods provided by the project management method. The guidelines for control of project management as presented in (Wijnen et al., 1988) were not discussed in the audit nor in the project documents. There was also no advise given in the audit report for improving the skills of the members of the teams individually. Yet it was mentioned that subject matter experts were not the most suitable persons to carry out function and task analysis. It was advised that senior heads of groups of BMB's should be asked to assist the work of the SME's, because of their responsibility for the functioning of the BMB's.

5.6.1 CONTROL OF TIME
The planning was made and presented in units of 1 week. Personal contributions were made in units of 4 man hours. All phases were marked with milestones. Phase 0 was performed before the formal start of the project. In phase 1 some subject matter experts, of some product groups spent less hours than planned in the schedule, for some less than 50%. The project leader demanded that time should be made for full commitment to the project. As the project was partitioned from phase 1 on, into parallel projects for different modules the delay in designing of the modules did not influence the progress of the others. After phase 1 the time scheduling became more informal and no general planning of the contribution of all team members by the project manager was made, however, activity planning was continued and done on a week to week schedule. Thus time planning was raised to another level: activities to be completed in time. The activities were described in text with instructions on how to perform them.

5.6.2 CONTROL OF BUDGETS
No budget planning based on realistic costs of man hours was reported for the internally performed activities of the project. During the later phases external parties were hired to perform design and technical activities, media selection, video script designs, video productions, textbook productions, etc. The costs of these activities were discussed and had to be approved by the steering committee.

5.6.3 CONTROL OF QUALITY
From the start of the project formal attention was paid to quality assurance concerning the products of analysis and design, e.g. learning objectives, learning contents and testing the learning results. Of course the modules were developed by different teams with other subject matter experts and to some extent sequentially. The TAM's for the modules developed first were evaluated by senior subject matter experts from outside the design team. Formative evaluation of selected parts of all the media was carried out using students from the target group.

5.6.4 CONTROL OF INFORMATION
During phase 0 standards were specified for documents and reports but not for storage and version control. For Phases 0 and 1 standard project documents were produced. No standard reports and documents were produced for the following phases. The control of information changed from formal to informal during the project. As the project manager achieved more control over the project he relinquished formal reporting by the teams. He wrote his own progress reports to be presented to the steering committee.

5.6.5 CONTROL OF ORGANIZATION
The method of Wijnen was followed for this way of control only during the first two phases. During phase 1 (definition) all tasks and activities were specified in general terms but they were not worked out in detail. The decision processes were also not worked out in detail. The audit after phase 1 produced the conclusion that the organisation was not optimal to motivate the teams and the individuals, this contributed to the decision to make the organisation for the next phases more informal.

The following assignments/partners were involved internally[1] (I) and externally (E), see

1. The amount of man power spent for the project internally was 1.6 man years.

table 5.11.

Partners	Department
Project leader and his staff (I)	Education and Training
Steering committee (I)	different
Quantitative need analysis (I)	Marketing Services
SME's (subject matter experts) (I)	different
Internal CBT-group -- programming (I)	CBT group
Media selection and learning objectives	external
Video script writers (I) and (E)	external
Video producers (E)	external
Textbook editors (I)	different SME's
Textbook printers (E)	external
Floppy disk duplicators (E)	external

Table 5.11: Internal (I) and external (E) partners involved in the project

5.7 Conclusions

The *BZD* course is sold on a large scale to local Rabobanks. It is expected that thousands of students will use the course completely or partly, in this respect the product of the *BZD* project is a success.

The project had many built in risks factors, it is therefore a good achievement that the project was successfully completed and that the product is used on a large scale.

Driessen (1993) evaluated the use of one chapter (Receiving) of the module Foreign Payments. This chapter is considered to be the most difficult of the whole course. The main goal of the evaluation was to measure how the course influences learning. He analyzed short questionnaires answered by 20 students and thinking aloud protocols of 6 students during their sessions on this chapter. With the questionnaires was measured *what* was learnt; with the protocols was measured *how* was learnt. The data of the questionnaires were analyzed statistically. The analysis gives some tentative general conclusions about the effectiveness of this chapter only. Driessen concludes that no significant learning is measured either after exercises or after simulations. Further he presents a list of remarks by users using the protocols. The most important remarks are those concerning the interactivity of the user interface:
- the simulation of the cases offers too less interaction with the student;
- the user interface could be more consistent;
- too less meaningful feedback on their results is given to the student;
- the commercial roles should be covered more explicitly;
- for certain subjects it is not sufficient to *teach* students. They have to be *persuaded* how to act.

These results have to be interpreted carefully because only one, but difficult and essential part of the course was analyzed. The form and structure of all educational software modules is congruent. This means that evaluation of other modules may yield comparable outcomes. In section 5.4.4 a part of one simulation from the module Credits is showed, see the tables 5.8-5.10 and figure 5.4. The interaction in this example is according to our opinion too "indirect", i.e. the student is forced to behave as a spectator of the semantics of the dialogue with a client. For this subject matter and for this target group of students this dialogue style is far too abstract and therefore not engaging and didactically not well chosen. This dialogue style is the same for all simulations in the course. This is supported by the conclusions of Driessen.

We will now answer the research questions for this project referring to the analysis described above. The implications of the answers are discussed briefly. See table 4.2 for the relationships between the questions and the aspects of the framework of Seligmann, Wijers and Sol (1989).

Question 1: How is the emphasis on instructional versus informational technology aspects divided?
This project was managed by the department Education and Training. In the way of thinking, see 5.3, strong emphasis was put on instruction technology. Much experience and expertise was available in this field, including experience in management of traditional instructional design projects. Within the development team, however, experience with professional design of multimedia educational software was modest and experience in the disciplines of systems design and software engineering was completely absent. This caused conservative use of the possibilities of the new media in the product. Under these conditions, an authoring language seems to be a good choice for the realization phase, however, it does not compensate for the lack of expertise. Hiring this expertise from an external partner and applying a more suited project management method would have improved the media use in the final product and its maintainability in the long term. After completion of the project the team was dissolved. No technical documentation except the source listing in the authoring language Tencore is available. Improvement of the product will require a complete educational and technical redesign.

Question 2: Is the product under development considered as a tool and/or as a medium?
This project shows strong emphasis on traditional instruction, not on motivation and engagement of the student with modern media. The video components of the course are semantically related to the contents of the educational software component but they have been designed and produced by external companies as rather isolated products. The form and the interaction of the educational software are not tuned to the form of the video components. In this project the available expertise on interaction design and media use was modest. The educational software does not have high and attractive interactivity. The formal style of communication within the banking world is, however, practised consistently for all the interactions built into the course. Educational software is considered as a tool, not as a medium. The final product screen layout looks fine but this is cosmetic. Most interaction is text-based including that during the "simulations" of dialogues with business clients at the counter. This kind of educational software requires more media approach and more realistic simulation of dialogues than the "multiple choice" dialogues provided. A summative evaluation (Driessen, 1993) of one module of the *BZD* course reveals that the students are not satisfied with respect to interaction and engagement, although the cosmetics and the layout of the user interface is fine. They felt that they are taught, not persuaded during simulations and training.

Question 3: Are the macro, meso, and micro levels of modelling distinguished explicitly?
The first steps at the macro and meso levels of the sequence discussed in chapter 4 were followed very strictly. As the subject matter, dialogues and procedures between employee and client at the counter, were not known beforehand, the why and the what questions at the macro

and the meso levels had to be answered in fine detail. At these levels excellent models were made, see 5.4. The emphasis during this project was put to the macro and meso levels. The conceptual design of the user interface, last part of the meso level, was not discussed extensively, nor was the micro level. The semantic, syntactic and lexical steps for instruction and interaction design were not performed explicitly. It is not understood or mentioned that semantic, syntactic, and lexical design steps have to be handled by different disciplines. Less attention was paid to user interface design, not due to malfunction of the team but by not recognizing the critical importance of the user interface in educational software. The answer to this question is correlated with the answers to the previous questions 1 and 2. From these three answers it can be expected that the unique advantage of educational software, motivation by interaction and engagement, is not well achieved. Fortunately, educational software can have more advantages than motivating students, e.g. the availability of the course independent of location, time and instructors. A definite summative evaluation of the whole course can give an indication of the overall educational quality of the *BZD* product. The use of the product on a large scale was not caused by a proven educational quality, however, at this time the product seems to fulfil a demand from the target group.

Question 4: To what degree are different responsibilities within the development team defined and implemented?
This question concerns the way of working, analyzed in 5.5. As this is an in-house project the members of the development team were appointed from different departments of the Rabobank organization. Most of the departments were not used to co-operating and communicating for designing an educational product. Therefore it proved difficult to arrange strong commitment to well defined tasks for the employees from the different departments. The tasks and the responsibilities were not described in detail and formally agreed beforehand. The project leader and the steering committee could not effect their mandate during all phases of the development. This introduced extra risks for the project.

This question is important for the way of working. The film making approach for the way of working would have increased commitment from the team members. Hiring external experts, not only for small isolated jobs but also for general responsibilities, e.g. reviewing the user interface at the milestones of the project can help to increase commitment from all members. This project started with a list of responsibilities for the first phases but they were not implemented and project management became very informal later due to lack of commitment.

Question 5: To what degree is communication realized explicitly between different disciplines?
This question also concerns the way of working. The disciplines co-operating in this project were many: subject matter expertise from departments within Rabobank with rather different products and market policies, project management from the project leader and his staff at the department of Education and Training, instructional design expertise, educational software expertise, etc. Communication between disciplines always needs special attention. Within this project the informal specification of tasks and responsibilities did not provide support to bridge the "cultural" differences of the disciplines involved. Representatives from several disciplines did not have the same aims and goals. Communication with external partners from different disciplines went better because the relationship was more formal and clear and because the aim to deliver good products was more explicit. During this project no formal engineering disciplines were involved. So, communication with and a contribution from engineering was almost absent. The management method used did not support communication and co-operation

between disciplines explicitly. So, the communication problem was not anticipated and supported. Use of a QA-plan according to the framework of chapter 4 would have improved communication between team members considerably during the project.

Question 6: To what degree are the design and realization of the product separated into teams with different skills?
This question also concerns the way of working. The educational design was organized carefully, as a result of the way of thinking, however, realization was organized rather informally without a technical design phase. No approved scenario or script of the design was available and forwarded from the design team to the realization team. Separation was informal and influenced by different approaches on both sides. Indeed, design and realization were separated, but transference of thought was not optimal because the realization team had another opinion about educational software development. The educational design did not contain an explicit user interface design. It was decided to let the realization team specify the user interface according to the general standards for educational software at the Rabobank. These standards covered layout and general navigation mechanisms within educational software, not the motivating and engaging aspects which depend strongly on the subject matter, educational goals and the target group of students. So, design and realization were separated into teams with different skills but incomplete designs, e.g. missing complete user interface specifications, cannot be improved during the realization phase.

Question 7: Is the project directed by one person who has a clear image in mind of the characteristics of the final product?
This question concerns the way of controlling. The project leader was an experienced instruction designer and project manager but he did not have experience in multimedia educational software design nor he had expertise on the subject, banking. Therefore, it was difficult for him to play the role of director according to the film making model. Playing the role of director was also hampered by the fact that it was difficult to demand full commitment from subject matter experts from the commercial departments and from members of the internal CBT group, and from other departments within Rabobank. Some conflicting interests were perceived between line and project functions. The project leader was conscious that the role of director is important but he lacked the facilities and support to implement the role according to his wishes. The answer to this question is a negative.

Question 8: To what degree are explicit quality assurance measures provided?
During the educational design phase when expertise within the team was sufficient, strong management support was available. The documents on management, budgets and schedules were written and approved according to the management method chosen for the project. The method supported the general definition and problem analysis phases well. The method was not, however, suited to the engineering phases. The project management, therefore, changed the management style into a less formal style at the very moment that engineering and the other non-educational disciplines became involved in the project. Only the assignments given to external parties were controlled by the formal project management of the project leader. The management method used was too general and not suited for (educational) software development. The quality assurance guidelines were too general and abstract with examples from traditional management projects not concerning any aspects of information technology. The *BZD* project was severely impeded by this.

6. CASE 2: AIRCRAFT MAINTENANCE TRAINING

This case study was carried out at an aircraft manufacturer. Computer based training is a well known phenomena for flight and other simulators. The world of aircraft manufacturing and operation is well known for its strong emphasis on security, safety, and quality in general, and (inter)national agreements and rules to guarantee and maintain these aspects. This case was selected for the following reasons. First, educational software is not so new in aircraft maintenance (Richardson, 1984), so experience in developing and using can be expected in this sector. Second, this was an *out-house* project with internal and external project management and with a considerable amount of work done by external contractors. Third, the product was accepted and finished for use on a large scale, 1000 or more students. Fourth, use of the product is compulsory for employees of airlines who buy the particular aircraft. Fifth, the educational software produced for the project provided more than 10 hours of session time and was integrated with other media.

First the context of the project in general is described in 6.1. Then the characteristics of the educational product are given in 6.2. All available internal project documents were studied and the two project leaders were interviewed. In sections 6.3-6.6 the development process is analyzed using the framework presented in chapter 2. Finally, the research questions are answered in 6.7.

6.1 Fokker Aircraft and maintenance training

Fokker Aircraft is one of a small number of European aircraft manufacturers. The company was founded in 1919 by Anthony Fokker, the successful Dutch pioneer in aircraft production. Today Fokker Aircraft produce the Fokker 50, Fokker 70 and Fokker 100. More than 250 Fokker 100's have been sold since 1988 and more than 100 options have been taken. The Fokker 100 is sold in numbers ranging from a single aircraft upto 75 to about 25 customers at the time.

When an aircraft is sold maintenance training (MT) also has to be provided to the customers. Maintenance can be divided into two types: *prescribed* and *troubleshooting* maintenance. Prescribed maintenance can be classified into three categories: scheduled or preventive maintenance, unscheduled or corrective maintenance and preparation for commercial aircraft operation. The prescribed maintenance tasks consist of procedures that are described in detail in the official aircraft maintenance manuals (AMM) belonging to an aircraft. Scheduled maintenance consists of lubrication and servicing, operational checks, functional checks, inspection, restoration. Unscheduled maintenance tasks are carried out in accordance with the following, sequential steps: verification of the fault, diagnostic task, remedial task, verification. Preparation for commercial aircraft operation does not belong fundamentally to technical maintenance. In reality maintenance technicians are also trained for these jobs, for example: cleaning, de-icing, protection against sand or dust, fueling and defueling. Scheduled and unscheduled maintenance can be carried out in line and in hanger. Line maintenance consists of regular control for all flights and at the end of each day. It is performed mostly on the turmac immediately between two flights. Diagnosis and small repairs are done using standard procedures and testing devices. When serious problems are discovered the aircraft is handed over to other technical employees who are able to dismantle parts, for unscheduled maintenance. A different, particular kind of scheduled maintenance is the extensive control and revision of an aircraft after a prescribed number of flight cycles. This kind of maintenance is carried out by engineers rather than technicians and is not subject of this study.

The courses provided by Fokker are line maintenance courses tailor-made for a specific version of the Fokker 100. They are provided in three different forms. In all forms Fokker is responsible for the training materials corresponding exactly to the relevant version of aircraft. MT can be given in the following forms:
1. at the Fokker MT Center at Schiphol Airport by Fokker instructors to student technicians from the customer (at present more than 26,000 student technicians for all types of aircraft);
2. at the training centre of a customer (an airline) by Fokker instructors to student technicians (sometimes future instructors when the customer is preparing for form 3) of the customer;
3. at the training centre of a customer by company training instructors trained by Fokker.

Three different MT type conversion courses[1] are offered by Fokker. Each course can be followed independently. Two of the courses are also offered as one integrated course. The three types of conversion courses for the F100 aircraft are:
1. airframe and power plant (29 days, courses 1 and 2 integrated take 33 days);
2. electrical system and (27 days);
3. avionics (23 days).

Traditional maintenance training at Fokker consists of a combination of classroom training, hands on practice on the Fixed Base Simulator (FBS) and visits to the production line. The FBS and the production line are only available at Schiphol Airport. Each course takes several weeks of full-time training. Each day an average of 6 hours classroom training is given. In the traditional form of the courses every day the students have two hours of practical training alternating between the production line or the FBS for hands on training. All courses start by teaching Description and Operation (D&O) including the system schematics of the aircraft. Then comes Maintenance Information (MI). The maintenance procedures are taught using the ATA[2] structure (or ATA chapters) for subject and systems of aircraft. A subset only of the ATA structure is supported by these MT courses. On request, each course can be taught in a more or less condensed way to shorten the duration of the course. The students always have experience in maintenance of aircraft other than the Fokker 100 covered by the course. They have already finished basic courses on general aircraft maintenance. After this course a certificate is issued to the students. A student can fail the course and therefore fail to be granted certification. Thus, all Fokker 100 MT courses can be considered to be a type of conversion course yielding additional certificates and with them certification for specific aspects of aircraft maintenance.

Another typical aspect of maintenance training is the requirement for aircraft safety. The maintenance technicians have to do their work quickly, under strong time limitations and in stressful circumstances. At the same time they have to do their work with an almost 100% guarantee of perfection. To achieve such high perfection (inter)national regulations and certifications have been drawn up. One international regulation is that subject matter in all MT courses should be structured according to the Air Transport Association of America protocol. In this protocol all possible maintenance tasks are given a number and a name and they are listed hierarchically with tasks and subtasks.

1. Seven other courses offered by the same department are not discussed here.
2. ATA: Air Transport Association of America.

The courses are presented by instructors, organized into three groups, see figure 6.1. All courses are developed (tailor-made) by a team of instructors in the form of a *lesson plan* for the individual instructors and a *training manual* for the students. When Fokker 100's are sold to a new airline the MT courses have to be adjusted because most new aircraft have slight differences due to new technological developments at Fokker Aircraft or to the special requirements of the customer. Of course the contents of the MT courses must always correspond exactly to the version of the aircraft. The teams of instructors have to base new lesson plans and training manuals upon the corresponding official Maintenance Manual for the aircraft and other resources. During presentation of a course the instructor must follow this lesson plan and use the prescribed set of slides, but he is free to work out the subject matter according to his own didactic style.

The Maintenance Training department of Fokker Aircraft is responsible for the development and provision all MT courses. The department consists of three independent teams of instructors (about 25 instructors in total). Internal support is provided by about ten employees to produce Training Materials and about five employees for Operation and Control. During the computer-based training (CBT) project described in this study a CBT project group of three persons was created at the MT department. The department is one of the departments belonging to the division Product Support, the departments Maintenance & Engineering and Technical Publications are also relevant. Other divisions relevant to MT are Engineering and Manufacturing, see figure 6.1

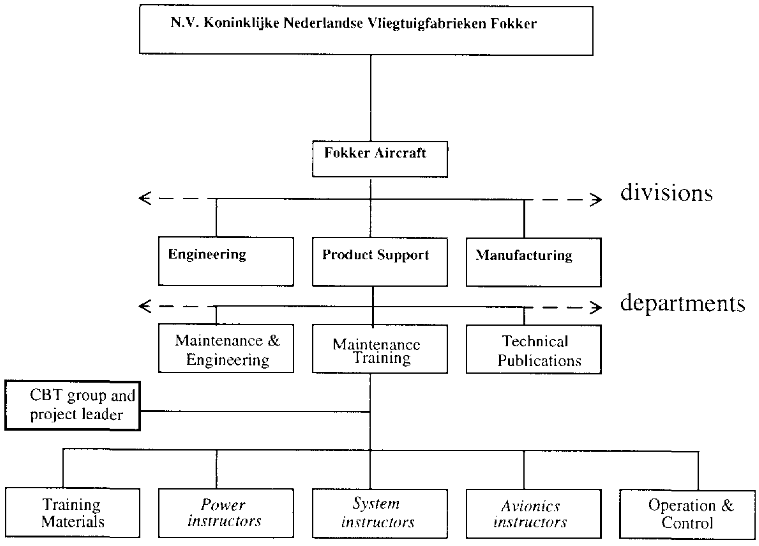


Figure 6.1 Organization of Fokker Aircraft showing the department Maintenance Training involved in the Fokker 100 MT project. During this project the CBT project group was considered to be an external partner of the department MT since its members were drawn from other divisions of Fokker.

The MT department decided to develop, for the first time, a comprehensive and strategic CBT-package for several reasons. As a full time traditional training department there was neither capacity nor expertise in educational and multimedia technology to develop educational software in house. Therefore it was decided to develop CBT in co-operation with an external partner having the necessary expertise and strong experience with management of large projects, to guarantee results within time and budget.

The CBT project analyzed here is a typical out-house project with all subject matter expertise available in house and with a tradition of striving for highest quality in the small management team of the department of the problem owner. BSO/Origin[1] was chosen as external partner. BSO/Origin is an international software house with a business unit for instruction technology and multimedia in the Netherlands.

6.2 The Fokker MT project (*Fokker MT*)

6.2.1 WHY CBT?

Maintenance training for aircraft technicians depends strongly on the availability of aircraft. It seems to be obvious that by simulation of subsystems of the aircraft the need for real aircraft for training purposes can be decreased significantly. The problem is the same as that of training pilots. If initial training can be done with simulation (cf. flight simulator for pilot) the costs of training can be reduced without any loss of quality. Orlansky and String (1983) showed that maintenance simulators for military training can be cost effective. Costs can be decreased by up to 40%, the simulators (standard, non-standard and CBT simulators) are as effective as traditional equipment trainers when measured by student achievement at school and there is no difference in job performance.

Fletcher (1989) completed a quantitative meta analysis on interactive videodisc instruction applied in defense training and in the related settings of industrial training and higher education. Overall, interactive videodisc instruction demonstrated sufficient utility in terms of effectiveness, cost and acceptance.

Richardson (1984) reports on computer based simulation in support of avionics maintenance training, utilizing a videodisc picture database to represent the actual equipment. It is necessary to shift the focus of simulations from equipment to the task. To achieve this the developers have to apply a method for on-line task analysis, as proposed by Richardson. This method should be suited for use by subject matter experts.

It is assumed that the availability of a non operational aircraft on the ground for training purposes is demanding, such a Fokker 100 aircraft is worth more than twenty million DFL. With a rate of 10 training hours a day over 20 years it is estimated to cost more than 2000 DFL per hour. With a maximum of two students working simultaneously on the aircraft (Van Beckum, 1992c) this equals some 1000 DFL per student hour. See, for some international papers on this project, Van Beckum (1992a, 1992b)

After selling the first series of Fokker 100, Fokker Aircraft carried out a successful feasibility study on the development of CBT materials in 1989 to improve both the quality and the efficiency of maintenance training for the Fokker 100. The quality of the courses could be improved as from extensive experience with traditional MT courses it has been reported that the amount of hands on training is insufficient. As a result some students are uncertain about the way they have to perform important tasks. They are afraid of touching the wrong switches. In the old situation students also had to wait during the training on the FBS, because only two can work at the same time. Traditional maintenance teaching consists of classroom training, hands on practice on the FBS and visits to the production line. After following the MT courses

1. At the time of the Fokker 100 CBT project its corporate name was BSO and the participating business units were named as BSO/Instruction Technology at Baarn and BSO/CAT at Schellinkhout.

at Fokker MT Center the students have to acquire on the job training at their airline, see figure 6.2a. The aim was to decrease the need for on the job training by adding exercises using CBT during the courses at Schiphol, see figure 6.2b.

After this feasibility study the first customer asking for CBT was American Airlines (AA). AA required 30 hours[1] CBT added to the classroom materials for delivering Fokker 100 MT courses in the USA under AA responsibility. This order triggered the large scale CBT project studied here. The main aim of AA was to improve cost effectiveness of MT by integrating CBT. The aim of Fokker was the same, but in the different context of the MT Center where the FBS and the production line are still available for use[2] by student technicians.

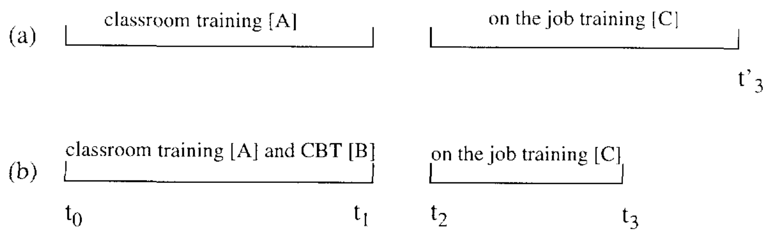


Figure 6.2 Decreasing the need for supplementary on the job training [C] at the airline by augmenting classroom training [A] (including FBS practice and visits to the production line) with CBT [B]. This results in t_3 being less than t'_3.

The CBT was intended for use by Fokker to train aircraft technicians from all airlines at the MT centre at Schiphol. The CBT is also suited to be sold to companies owning large numbers of Fokker 100's. They can use the CBT for training their maintenance employees at their own training center and on site at the airports where the aircraft are maintained. Another argument for developing CBT is the general trend to provide multimedia CBT courses by competitors in the aircraft market.

6.2.2 REQUIREMENTS

The learning objectives applicable to the courses, are described in the curriculum analysis as follows:
"After completing the training the student will be able to:
- locate and identify the main components;
- describe and operate the applicable systems;
- service the applicable systems in accordance with the line maintenance procedures;
- troubleshoot the applicable systems;
- test the aircraft and/or its systems after component replacement.

It can be concluded from the above that a significant part of the course must be directed towards the mastering of maintenance procedures."

1. Notice that a fixed number of maintenance procedures was not required, this was considered to be a dependent variable of the 30 required hours of CBT.
2. The number of sessions at the production line and FBS has not changed, only the content of the sessions.

CBT was intended to be a supplement to classroom training and FBS training. The ultimate goal was to develop a blend of training media: classroom, CBT and FBS training, resulting in increased effectiveness and efficiency of training.
The following points were considered to be the main benefits of the CBT course (initial requirements):
"1. Improved confidence of the students in what they are doing[1]. They should perform their tasks without much hesitation and uncertainty.
2. Improved performance of the procedures because students learn and practise to perform them correctly and within the necessary time limits.
3. Effective study during the waiting hours for the FBS, which will, in the future, influence the number of hours of classroom training."

The target population for the CBT consists of technicians or students each having the same characteristics as those on the regular courses without CBT. Sometimes the target group is a team of instructors from an airline that has a number of Fokker aircraft in use and that is preparing to provide MT as part of their responsibilities. The student technicians are usually licensed aircraft maintenance technicians. The students have completed: basic aircraft maintenance training (comparable to MTS level in the Netherlands), a type-rating course of several months and experience in component location and removal/installation, depending on the country the students are from this experience with other aircraft is between a few months and tens of years, before starting this course. Potential instructors are given an adapted course on the same subject matter to prepare them to teach the MT course at customer sites.

The training is considered to be the preparation for an official exam to become an aircraft technician with a Fokker 100 type rating which qualifies the technician to maintain the aircraft and to declare the aircraft technically airworthy.

The written (on screen) language used in the course should be "simplified English", the spoken (on audio) language not. The age of the students is in the range from 20 to 65. The motivation of the students is high. To provide extra motivation for the students the CBT should be challenging, interesting and stimulating. Although the maintenance technician has to work under time stress, e.g 30 minutes between flights and from 22:00 to 06:00 at night) this phenomena has not been built into the CBT[2].

The instructors of MT courses have to know how to apply this CBT program within their lesson plan. Therefore integration of the CBT exercises with the classroom teaching had to be considered carefully during the design phase.

Finally, we can summarize the following general requirements as described in the project documents:
1. The learning objectives are training procedures as covered by the traditional classroom training according to the ATA structure. Tests should be included within the CBT. The requirements do not specify *which* procedures have to be included. This is not part of the requirements;
2. A minimum of 30 hours of CBT must be delivered for American Airlines;
3. The first estimation of the number of procedures to be implemented is 72;

1. This can be measured by the airlines after a student technician has completed the course and by Fokker during exercises on the FBS.
2. Time stressing features were built into the course on several occasions. These were "switched off" in the final version because such artificial time stressing introduced some undesirable and unrealistic aspects into the simulation.

4. The natural written language should be "simplified English";
5. The instructors are responsible for the subject matter definition and for the acceptance procedure for the contents of the final CBT product;
6. The hardware platform is both MS Windows (for AA) and Apple Macintosh (for Fokker at the MT Center);
7. Maintenance of the product should be possible by staff of the Fokker MT department;
8. Technical project management and product development should be performed by an experienced partner.

6.2.3 THE PRODUCT

The subject matter of the *Fokker MT* project consists of many procedures to control and to repair functions and parts of the aircraft, see table 6.1 for a list of all procedures implemented. Each procedure is defined as a series of actions to be performed sequentially (i.e. step by step). Before the technician is allowed to maintain an aircraft he has to know some fundamentals concerning the functions he has to maintain and he must have practical skills.

00 PUBLICATIONS
Trouble shooting schematics manual
21 AIRCONDITIONING
Temperature control
22 AUTOFLIGHT
AFCAS maintenance panel
AFCAS fault isolation
Auto throttle operation
Auto pilot & flight director
23 COMMUNICATIONS
ACARS operation
ACARS maintenance
24 ELECTRICAL POWER
AC generation BITE (built-in test equipment)
Battery circuit
26 FIRE PROTECTION
APU fire extinguishing
Fire detection BITE
27 FLIGHT CONTROLS
Rudder servo-valve sticking circuit
Stall warning system
Flap asymmetry
Liftdumper control
Flap operation change-over
Horizontal stabilizer control
28 FUEL
Auto refueling
Fuel quantity BITE
Magnetic fuel level indication
Manual refueling
Defueling
29 HYDRAULIC POWER
Hydraulic leak-rate
30 ICE & RAIN PROTECTION
Wing and tail anti-icing
Windshield heating system
31 INDICATING/ RECORDING SYSTEMS
MFDS page selection
Flight warning system
Flight data recording system
Maintenance and test panels fundamentals
Proximity switching BITE
32 LANDING GEAR
Landing gear operation
33 LIGHTS
Emergency lights control
34 NAVIGATION
IRS operation
IRS test
FMS operation
FMS maintenance
Ground proximity warning system
Windshear detection and escape guidance
EFIS Maintenance and diagnostics
Weather radar control
36 PNEUMATICS
Bleed air system
Bleed air supply and control
Double wall duct-leak indication
TPC BITE
49 APU
APU BITE
APU start
APU data loading
76 ENGINE CONTROLS
Emergency shut down system
78 EXHAUST
Thrust reverser de- & reactivation
80 STARTING
Engine air starting system

Table 6.1: List of all procedures implemented in the course MT for Fokker 100. The numbered titles are the ATA-chapters

The instructional strategy of the CBT was based on three types of learning: programmed instruction (Guided module), drill and practice (Exercise module) and testing (Test module). Three modules are provided for each procedure to be taught.

In the Guided modules the training objectives are presented first followed by an overview of the procedure called the "advanced organizer". The preparation steps and the component locations are described. All steps needed to perform the procedure are then presented. Finally the student technician has to act on the screen by choosing, switching or tuning objects displayed as part of the simulated device. Audio and visual feedback is provided on the result of the actions by the student. In this way the descriptive part[1] of the subject matter is presented step by step. All steps of the procedure are displayed on the screen using the metaphor of a notebook. This emphasizes the existence of many steps in all maintenance procedures, as recommended[2] by Richardson (1984). This guided module prepares the student to do the drill-and-practice work.

In the Exercise Modules the student has to practice procedures. After presentation of the required steps the student has to perform these on the devices displayed on screen. After one failure the student has one opportunity to redo the step correctly. After a second failure the right action is presented and explained to the student. If the student makes serious errors the results are presented on the screen graphically and realistically. See figure 6.3 for the main course menu and figure 6.4 for an example of an exercise for the procedure FUEL.

In the Test Modules the students can, when they think they are ready to do so, demonstrate without any guidance and help how to perform the procedure. The results of the actions are scored. The scores are presented to the student at the end of the modules. A CMI program[3] is used to report the scores of all students to the instructors.

A few Trouble shooting modules are also provided. Trouble shooting enables the student to apply the procedures learned in a problem solving environment. This environment consists of a simulation. These modules cover one or more procedures within one ATA chapter. The students are provided with a *problem*, for example a fault report by the flight crew. The problem has to be solved by diagnosis. In this process the student can apply some of the procedures he may have learned in the prescribed maintenance part of that ATA chapter. With trouble shooting exercises the student has to decide which set of maintenance procedures is needed to analyse the reported fault.

The three courses cover subsets of the ATA chapters, see table 6.1. In the form as taught at Fokker MT Center, the course on airframe and power plant takes 6 weeks, the course on the electrical system $5^1/_2$ weeks and the course on avionics takes 5 weeks.

The main objectives of these courses are described in the project documents using the same terms for the applicable systems: the air frame and power plant, the electrical systems, and the avionics systems. After completing the training, the student will be able to:
- describe and operate the applicable systems;
- locate and identify the main components;
- service the applicable systems in accordance with the line maintenance procedures;
- trouble shoot the applicable systems;
- test the aircraft and/or its systems after component replacement.

1. The subject "description and operation" is presented in the classroom. The guided module provides a tutorial with some interaction on this subject matter.
2. The paper by Richardson was, at the time, not known at the Fokker MT Department.
3. This CMI module consists of some Authorware Professional modules developed by the supplier of the authoring system in Europe according to requirements given by the project team. These modules were changed and improved by the BSO/Origin team. The CMI modules enable Fokker Aircraft to deliver the F100 MT lessons to the student.

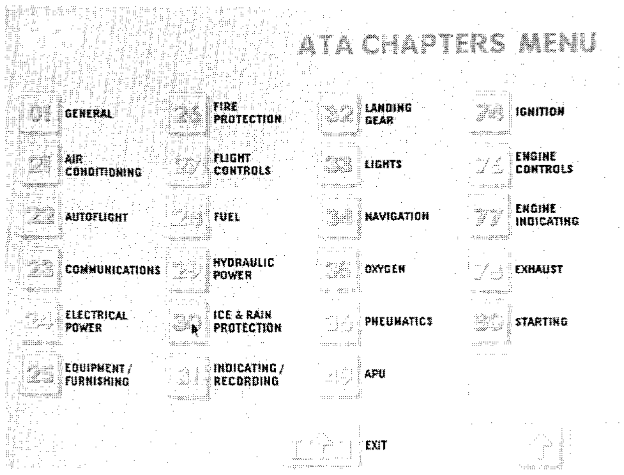


Figure 6.3 The main course menu presenting the ATA Chapters Menu.

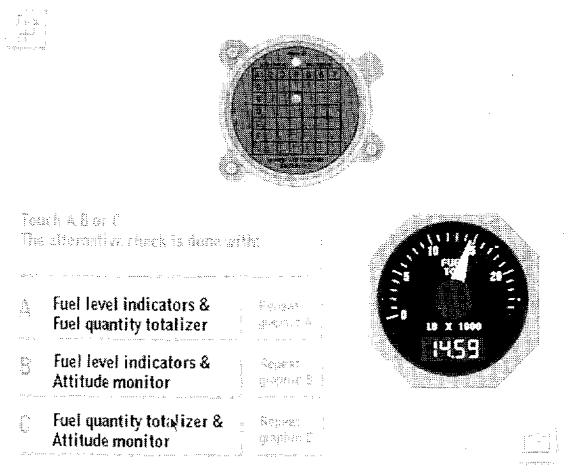

Touch A, B or C
The alternative check is done with:
A Fuel level indicators & Fuel quantity totalizer
B Fuel level indicators & Attitude monitor
C Fuel quantity totalizer & Attitude monitor
FUEL TOTAL
LB X 1000
1459

Figure 6.4 Screen of an exercise for one of the procedure of the ATA chapter *28. FUEL.*

The educational software consist of over 1 Gbytes of data and audio and 1 double sided videodisk containing video images. Over all, disciplines and partners together, approximately 30 man years[1] were spent to finish the CBT and some associated projects. After completion of this project the Fokker MT department was able to maintain the CBT independently. The department has already realized different versions of the CBT for other airlines and for other aircraft.

6.3 Way of thinking

This project is a typical out-house development project. The responsibility for the complete development of the educational software was given to a third party under very strict conditions. The "problem owner" of this project was the MT department of Fokker and its implicit and explicit ways of thinking were most important. The strategy and ambition of this department is the basis for the project and the involvement of BSO/Origin in it as a third party. BSO/Origin spent more man hours in designing and realizing the software and the media than Fokker MT did, but the latter defined the general goals and criteria to develop the educational software, to implement it, to strive for high quality and finally to acquire the expertise to modify and maintain the product after the end of the project. Maintaining the educational software is a strategic concept for the product because the Fokker 100 will be delivered in different versions to different customers. MT courses must match exactly the maintenance of the version of aircraft in use by the operator the student technicians are coming from.

As the characteristics of the subject matter and the international students were very well known beforehand it was possible to be sure about the "market" and "use" of the multimedia to be developed. The general trend that many aircraft manufacturers are increasingly providing educational software for MT and that, hence, many airlines are demanding educational software for their MT programs, also gave emphasis to a strong positive vision about multimedia educational software for MT at Fokker.

The core of the MT department consists of about 25 instructors each one of whom is an excellent subject matter expert. The internal organization and management of the MT department is rather informal but shows an extreme policy to achieve the highest quality in traditional MT. The instructors, however, have no explicit expertise on didactics and educational technology, nor about educational software development, and software development in general. They did not have much capacity, because they had to give regular MT courses continuously during the course of this project, to contribute to the design of the educational software. The management of the MT department had only a limited capacity to participate in the daily management of the project. All the factors mentioned above caused Fokker MT to call in BSO/Origin as a party able to supply the expertise and the capacity for the project. Fokker decided to suffice with a strong dedication to quality control of the project within a small CBT group and a steering committee with representatives of Fokker MT and BSO/Origin. The head of Fokker MT brought in a strong vision about the results required for the project. The project leader on behalf of Fokker MT brought in a strong commitment to assure the overall quality of the results and the budgets[1] and schedules to achieve this. The responsibility for most phases of the project was delegated to BSO/Origin. Fokker MT required that all BSO/Origin staff involved in the project had to work in house at the Fokker MT Center. During most phases about 10 BSO/Origin employees were working at the MT Centre.

BSO/Origin has high standards[2] for quality assurance for projects. They are skilled in developing educational software using the comprehensive method HOEP, see 2.2.7 and

1. About 12 man years by Fokker and about 18 man years by BSO/Origin, including conversion of all procedures from Macintosh to MS Windows and many additional supporting services by BSO/Origin. In this amount is also included the production of a set of manuals for developing new versions of the courses and a workshop for the CBT group of Fokker on how to apply these manuals. As a result the small CBT group proved to be able to develop new versions independently from BSO/Origin.
1. Both budgets for subject matter experts and for BSO/Origin.
2. BSO/Origin carried out several large projects for Fokker EDP departments. The Fokker MT Department knew about this experience which qualified BSO/Origin at Fokker as "having high standards for QA".

Hartemink (1988). Other departments of Fokker Aircraft already had good experiences with the informal but professional style of BSO/Origin during different types of software projects. Fokker expected from BSO/Origin that it would not only provide technically good solutions but that it was also able to accomplish this strategic project within a fixed time schedule and within given budgets.

6.4 Way of modelling

As the HOEP development method for educational software was used much attention was paid to modelling at all levels of design. The main modelling concepts during design are presented in figure 6.5. How the modelling concepts were used in context is described in the next paragraphs. Note that all modelling was carried out by the BSO/Origin team members. When necessary they asked the subject matter experts from Fokker to fill-in or work-out the contents of the models.

1. Text to specify the subject matter;
2. Composition diagrams to describe the structure of the educational software;
3. Flow charts to describe the flow of the dialogues;
4. Visualization and user interface proposals for the static part of the dialogue
5. State transition diagrams for the dynamics of the dialogue.

Figure 6.5 Modelling techniques during design. These techniques were used during the start up phase and during development of the first 4 procedures. Later, composition diagrams and state transition diagrams were no longer made.

6.4.1 SUBJECT MATTER

The subject matter was not the same as in the former, traditional maintenance training courses, and CBT is an essential part of all exercises. The training manual was directly "derived" from the aircraft maintenance manual by the instructors team. The CBT was not derived from this maintenance manual but again directly from the aircraft maintenance manual delivered by the Engineering department. The subject matter was ordered in ATA chapters, see table 6.1. Procedures were taught sequentially in related groups. The following teaching steps were followed for each procedure:

- step 1 (classroom teaching) Description and Operation of the procedure including:
    - objective;
    - function;
    - operation;
    - control (normal, abnormal, emergency);
    - location;
- step 2 (classroom teaching) Maintenance Information of the procedure including:
    - servicing;
    - testing;
    - special tools;
    - safe working;
- step 3 Hands on training on the FBS for the procedure;
- step 4 Visits to the production line relevant to the procedure;
- step 5 CBT on some simulated examples of the procedure.

6.4.2 TASK ANALYSIS

The maintenance tasks consist of procedures that are described in detail in the aircraft maintenance manual. As described in the curriculum analysis report of the project the tasks can be classified into three categories: scheduled tasks, unscheduled tasks and preparation for commercial aircraft operation (see 6.1). The content of the aircraft maintenance manual was filtered and ordered in the curriculum analysis report.

6.4.3 EDUCATIONAL DESIGN

Prescribed maintenance consists of procedures that are described in the existing manuals. The use of programmed instruction and drill and practise is suitable to teach such procedures. The CBT consists of at least three modules for each maintenance procedure: a module in which procedures are taught to the student technician, a module in which the student can practise a procedure and a module to test whether the student can perform the procedure. It is up to the course manager of any MT department at Fokker or an airline whether or not a student is required to follow all the modules. Depending on his experience and motivation the student can skip modules and go directly to the test module. The educational design of the standard module of this type is described in a top down way using composition diagrams, see figure 6.6.

Trouble shooting enables a student to apply the procedures he learned in a problem solving environment. High order problem solving skills are involved. Through dialogues and a simulation the student can find out what is wrong and how he can correct it. Every step he takes can have an effect on the fault indication and the consequences can be simulated on screen. The design of this kind of training modules can also be described using composition diagrams, see figure 6.6.

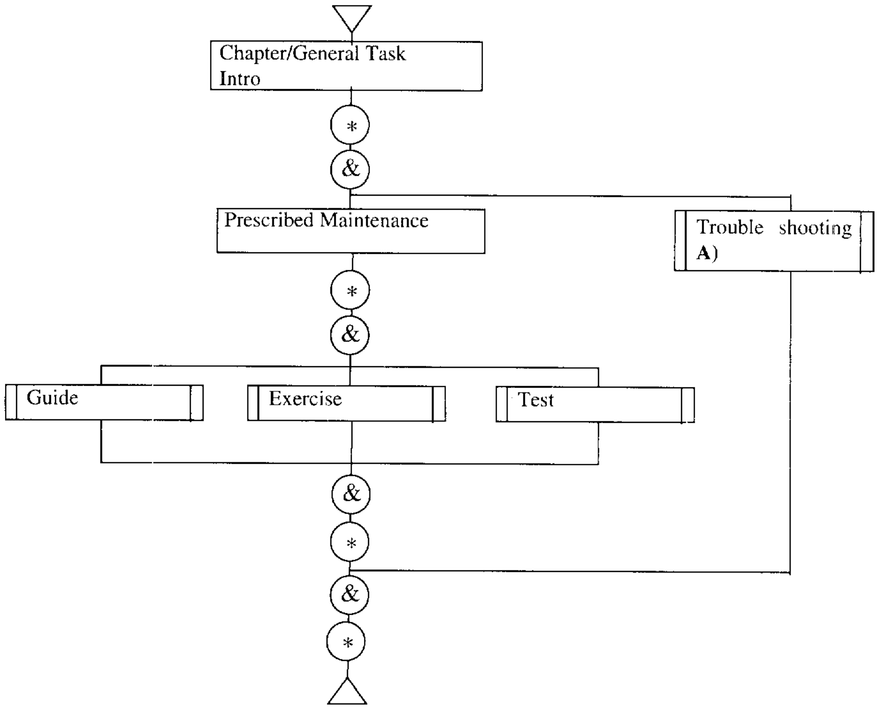


Figure 6.6 Composition Diagram describing the outline of an ATA chapter.
Legend: * indicates repetition until student wants to stop;
& indicates the student can do all tasks at the same time or at the same level.
A) A trouble shooting module was developed for only a few chapters.

Learning to trouble shoot was not the main aim of the courses. The trouble shooting module is not implemented per procedure but per particular ATA chapter.
Design principles for "good" CBT interaction are used for two types of brakes: brakes planned into the CBT program and interruptions, e.g. asking for a coffee break or placing a bookmark, of the planned flow by the student. Both types are used in the product.

6.4.4 INTERACTION DESIGN

This general design of the user interface can be considered to be the conceptual design, see chapter 4. The metaphors used and the concepts and objects to be presented to the student, e.g. menus, notebook, simulations, tests, are already defined. The functional specifications of the user interface were worked out in a separate document. This is the semantic design according to chapter 4. This was all described in text. The syntactic design was described using state transition diagrams, see figure 6.7. These diagrams were made for the design of the first four procedures, the pre-production phase. For later procedures the state transition diagrams were omitted to save time and because the subject matter experts had difficulties with reading these diagrams. State transition diagrams did not improve the communication between the subject matter experts and the educational designers (specificators[1]).

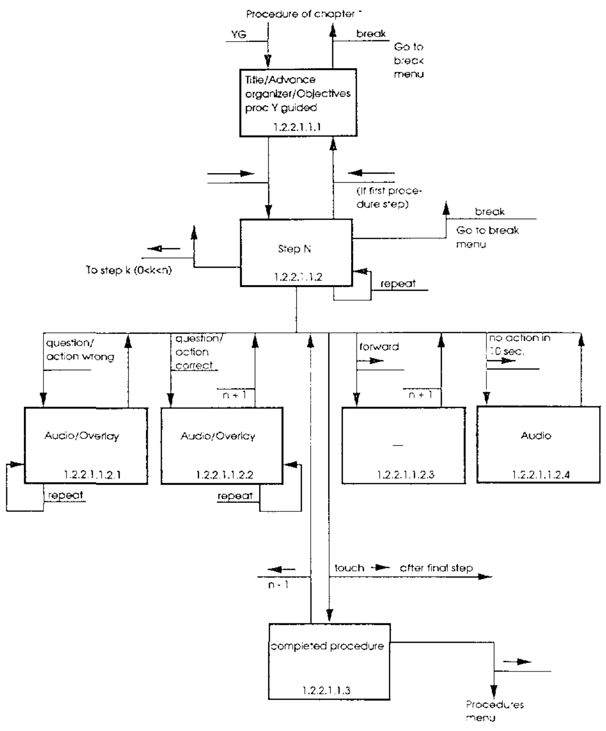


Figure 6.7 Standard State Transition Diagram for the Guided Module

1. In the project documents of this case and in the QA-plan the term *specificator* is used for educational designer.

Interaction specification forms were used for the semantic design of the screens, see figure 6.8. The lexical design of the screens was done on paper with visualization proposals.

BUT*-interaction
Model: BUT8(6)
Pre-erase: Yes
Graphic: Schematic of front view of aircraft; you can see body, collector tanks and two wings.
Audio: What is the location of the magnetic fuel level indicators? Point to them.
Score: 3
Intact: Push the six positions of the fuel level indicators
(correct)
Audio: Correct
Overlay: (After pushing each single position) Show indicator in colour.
Intact: none
(incorrect, not pushed)
Audio: You have not pointed to the positions of all six magnetic indicators
Highlight: Positions not yet pushed
Intact: Positions not yet pushed
(incorrect button pushed first time)
standard feedback
(incorrect button pushed second time)
Audio: The positions are these ones. Point to them.
Highlight: Positions indicators
Intact: Positions indicators

Figure 6.8 Example of semantic specification of an instance of pushing on button of type 6 as defined by the form printed in bold.

6.5 Way of working

The way of working was based on the HOEP method described in section 2.2.7. This project has some special characteristics. The subject matter was defined on paper reasonably well beforehand[1]. The educational software to be produced consisted of more than 50 procedures with a similar, but very complex structure. This structure had to be designed first. Many small but different audio, video and graphic components had to be designed, realized and integrated for each procedure. Developing all procedures sequentially is not efficient. The planning should be organized in such a way that all disciplines stay in work continuously: subject matter experts, educational designers, audio and video specialists, programmers, etc. This was solved in the following way.

During a "pre-production" phase the design and realization of the first four procedures was done simultaneously. The purpose was to use the work on these four procedures as a way to find good estimates for the man hours needed for the rest of the project and to try out the hardware and software tools; to find the best way of technical working and to train the development team. The pre-production phase also included curriculum analysis, instructional design, selection of tools, creating the technical infrastructure and specifying design standards. This was a creative, difficult and critical stage of the project because most activities were first

1. The F100 aircraft was under development during this project for developing educational software. Thus, not all details of the aircraft were already known to the CBT development team from documents. Some details were communicated directly from the engineering department.

attempts without clear feedback and results. This phase could also be considered as prototyping to find the right concepts, solutions, formats and tools to proceed and to succeed with the project.

The "procedure" is the unit of production. To improve the efficiency of the way of working the "sub project" was chosen as the unit for planning. The project as a whole was divided into three sub projects:

I.	16 procedures (first 4 were developed in the "pre-production-phase")
II.	18 procedures
III.	17 procedures

After each sub project the method guidelines, the different standards and the quality assurance plan were adjusted and improved.

Per sub project the following phases were planned: Inventory phase by Fokker, Specification phase, Realization phase, Test phase, Revision phase and Acceptance phase. An overview of the phases is depicted in figure 6.9.

1. feasibility study
2. Start-up phase (containing pre-production phase for 4 procedures)

3. Inventory phase 4. Specification phase 5. Realization phase 6. Test phase 7. Acceptance phase	production phase per procedure (47 procs)

8. Delivery phase
9. Use and maintenance

Figure 6.9 Project phasing per procedure according to (Hartemink, 1988)
The first four procedures were developed using the same phases (3-7) during the start-up phase (2).

Two documents were produced during the start up phase: the general educational design and the project planning, including quality assurance.

The general educational design consisted of:
1. Curriculum analysis[1]: problem definition, target group and subject matter analysis;
2. Infrastructure: educational context, implementation context, technical infrastructure, distribution and copyrights;
3. Preliminary solution: instructional strategy, global structure and global contents;
4. Financial planning: cost analysis and project definition.

The project plan consisted of:
1. Project organization: steering committee, project team, task definitions, responsibilities, skills and location of the project;
2. Phasing: following HOEP;
3. Progress reports;
4. Standards and tools;
5. Planning: general financial planning, activity scheduling, detailed planning for the

1. The term "curriculum analysis" is used in the project documents written in English.

specification phase;
6. Financial contracts between Fokker and BSO/Origin.

The phases 3-7 (figure 6.9) are carried out per procedure. The inventory phase was performed by subject matter experts from Fokker. The specification phase was performed by a "specificator", one SME and one member of the CBT group of BSO/Origin. The realization phase was performed by programmers of BSO/Origin assisted by a co-ordinating member of the CBT team of BSO/Origin. The test phase was performed by programmers and specificators of BSO/Origin. The acceptance phase was performed by SME's and a member of the CBT group of Fokker. The SME's, specificators and programmers all represented different disciplines. During the first iterations scheduling the contributions by SME's and specificators prove to be difficult because of the limited availability of the SME's for the project. The SME's have to contribute heavily to the regular training program. The productivity of the specificator and the programmer assigned to one procedure was decreased by this scheduling problem. The a solution was to develop many procedures simultaneously and to train programmers to perform the specification. This increased the productivity of the whole iteration. The quality of the design, however, detoriated slightly as a result of this measure, derived from an increase in revisions needed, but, after a few iterations quality improved again. Another problem came up with quality control. When the same person does both the specification and the programming responsibility for the results of the phases became indistinct[1].

The phases of the production phase, per procedure, can be summarized as follows:
During the Inventory phase a team of SME's selected the subject matter of one procedure carefully. Secondly one SME and one member of the CBT group (see figure 6.1) of Fokker described the training objectives for the modules for this procedure. Finally a subject matter expert described the contents, texts, videos, steps, in detail using a checklist made during the curriculum analysis.

During the specification phase a specificator of BSO/Origin, assisted by 1-2 SME's and one member of the CBT group, wrote a specification report using checklists and templates from the general educational design, part of the curriculum analysis. This report defined exactly how the subject matter should be presented to the student. Part of the report was a visualization proposal. The guided module, the exercise module and the test module were specified in this way. The Fokker team then reviewed the specification report formally. The specificator then had to change the specifications according to the results of the review. Informal contacts with the specificator were normal during this internal revision. A formal approach by the SME's was, however, necessary.

During the realization phase by BSO/Origin all the graphics were produced by graphic designers. The audio visual and the videodisk materials were produced for several procedures at once[2] and finally all components were integrated.

During the test and the revision phases the integrated version of the educational software for this procedure was improved iteratively within the BSO/Origin team. All problems were registered on "standard" problem report forms and solved subsequently.

During the acceptance phase the Fokker team, SME(s) and a member of the CBT group,

1. Not until the end of the project were some procedures specified and programmed by the same person. Having already developed many tens of procedures some team members were very experienced. The total number of employees involved at the end of the project was reduced by this measure.
2. Batch-wise and with quality control by a member of the team of BSO/Origin.

performed at least two sequential acceptance tests that resulted in formal problem registrations to be used by the BSO/Origin team for revision. Before the problems were solved the BSO/ Origin team estimated the required budget costs. Finally the procedure was approved by the unit managers and the project leader from Fokker.

This way of working was organized as depicted in figure 6.10.

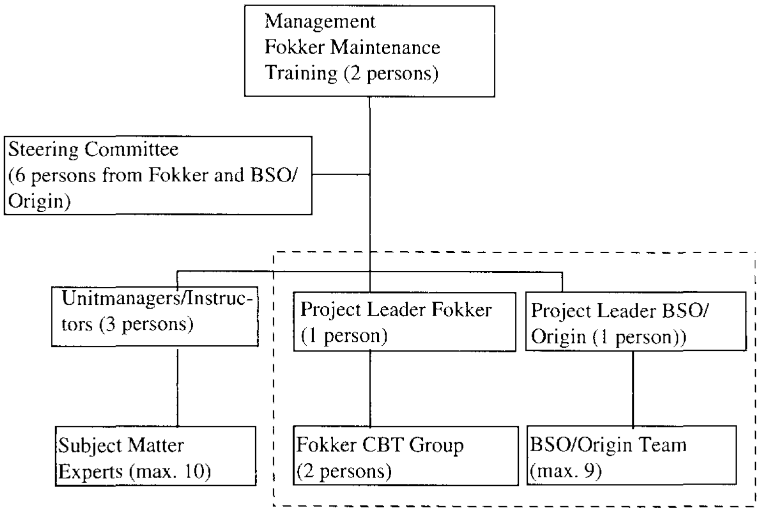


Figure 6.10 Project organization. The formal project team is depicted in the dotted rectangle. The unit managers and the SME's remained working for their line functions.

After sub projects I and II an investigation was done of the problem registrations made during the test phases, internal reviews within the BSO/Origin teams after integration of all realized media for one procedure, and the first and the second acceptance test by the CBT-group of Fokker, see figure 6.11 (Schouw, 1992).

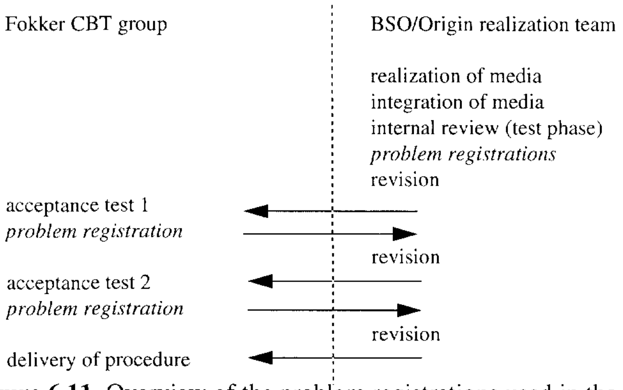


Figure 6.11 Overview of the problem registrations used in the study by Schouw (1992)

In total 1087 problem registrations and the times required to fix them were analyzed statistically. The conclusions of the analysis by Schouw are presented in table 6.2.

Significant result	Caused by
the number of problem registrations declined more than 25% from sub project I to sub project II	increasing experience within the team
strongest decline was reported with tests	increasing expertise within realization team
the most problems were registered during the test, the fewest were registered during acceptance test 1	the internal test within the realization team solved a lot of problems
specification problems were mostly caused by problems with contents (53%), not with interaction (34%) and instruction (13%)	misunderstanding between SME and specificator
realization problems of sub project II are more expensive to solve	SME's became more positive and creative concerning educational software
most realization problems concerned the screen (46%) and audio (26%), not video (8%) and software (20%)	abstract representation of subject matter was most difficult to design and to communicate between disciplines
During sub project I most problems were classified as specification problems. During sub project II as programming problems	SME's became more demanding and designed advanced features for educational software

Table 6.2: Significant results of the analysis of problem registrations by Schouw (1992)

The learning function paradox: "With increasing expertise the number of problems decreases, but the time to solve each problem increases" is presented in (Schouw, 1992).

The delivery phase (see figure 6.9) was a complicated one. BSO/Origin had to deliver products to Fokker MT and Fokker MT had to deliver products to American Airlines. Early on during this project the first plans were made to prepare a procedure and schedule for providing distribution, management and maintenance of the educational software for American Airlines and other airlines. At the end of the project Fokker MT expected to be capable of maintaining the product and to develop new versions for other versions of the aircraft and for other customers. A manual was delivered for members of the Fokker CBT team on how to maintain the CBT, including all phases of development at the procedure level as applied during the project. Training the Fokker CBT group by BSO/Origin was also part of the project. The CBT package was developed for the Macintosh. All material was converted to MS Windows during the project for American Airlines.

6.6 Way of controlling

As part of the HOEP method a quality assurance plan was made during the start up phase. This plan was updated several times during the project. Fresh experiences of the development team were incorporated in the new versions of this plan. Topics such as activities during all phases, templates for specification, standards for reviews and tests, programming standards and standards for design and realization of audio and video were described in the QA-plan. A general definition and guidelines were made beforehand for the acceptance procedures and these were refined during the project. Many forms were designed and used, see e.g. the problem registration forms mentioned in 6.5. The strategy for scoring in the exercise and test modules was specified. The acceptance process for the first maintenance procedures contained less iterations from BSO/Origin to Fokker and back again. During the project the number of iterations and the emphasis on good tests was increased, for some procedures upto 6. The acceptance process was underestimated and the right way of controlling was mastered during the project.

As one of the aims of the project was to deliver 30 hours of educational software to American Airlines it was extremely crucial to have a good definition of how to measure the session time of educational software. A protocol was specified for reading all screens carefully, as students are supposed to do, giving mostly the right and sometimes a wrong answer, reading and listening, all feedback, etc. In the subsequent versions of the QA-plan the responsibilities of all partners and individuals in the project were described, for the steering committee, for the project leader of Fokker, for the project leader of BSO/Origin, the Fokker CBT team, and the BSO/Origin project team. Standards were described for paper and electronic documents on texts, audio, video, graphics, animations and many kinds of user documentation. The hardware and software was described carefully i.e. the version of Authorware Professional to be used, drawing packages, word processing, sound editor, etc.

The decision to give feedback using the spoken word introduced high costs and extra logistics because all the intermediate revisions of the spoken text required new recordings of the text by the same professional readers or voices. These and other aspects required an explicit and strong emphasis on quality control during all phases of the project.

6.7 Conclusions

The Fokker 100 Maintenance Training software was delivered successfully to American Airlines in an MS Windows version, and it is being used on a large scale. Other versions for use on Macintosh at the MT Center at Schiphol Airport were successfully created by the CBT group of Fokker. Versions for other types of aircraft are being prepared by the same group.

First, after the pre-production phase, see figure 6.9, a formative evaluation was carried out with 5 instructors, and 6 students from the target group. Three modules[1] were evaluated by observing student behavior and questionnaires. The learning results were not tested explicitly, because test modules were built into all modules evaluated. Thus, finishing a module means having passed the built-in test. The outcome of this evaluation was that the learning time was as expected and that opinions of instructors and students were positive on user interface, suitability for the target group, and educational quality. Critical comments were given on the technical quality of the audio feedback and on many minor lexical details of the user interface. These details of the user interface were improved before the production phase was started.

At the end of the project the BSO/Origin team evaluated the project using structured interviews of all members of the teams[2]. Some of the relevant conclusions are given here:

1. All parties including the problem owner and the instructors judged the project as successful as far as:
- the quality of the educational software produced;
- the degree the project was controlled;
- the maintainability of the educational software.
2. The co-operation between the Fokker team and the BSO/Origin team was considered to be

1. The subject matter on AFCAS, fuel, and rudder was evaluated.
2. This evaluation is reported in an internal report.

positive by all individuals, however, both teams found the other team lacked knowledge of their discipline. Subject matter experts would have liked the specificators and programmers to know more about aircraft. On the other side the specificators would have liked the subject matter experts to know more about instruction technology and media. Direct communication on the subject matter proved to be difficult. To achieve more flexibility during the project borders between the disciplines tended to vanish. This resulted in decisions sometimes being made by the "wrong" people unknown to other relevant team members.
3. All assignments, responsibilities and competencies should be described more formally and recognized by all members of the teams;
4. The use of visualization standards was very important, but maintaining them was sometimes difficult; these standards should have more dynamics of the interaction included;
5. During the project a conflict was always present between the line function of the subject matter experts, who had to continue teaching in the traditional way and the project function of the other members of the team.

From these results can be concluded that, to our opinion, the overall quality of the final product is good. The problem owner of Fokker, including the instructors, and the customer American Airlines were content. They are all responsible for the use of the product as a medium to practice the maintenance procedures taught in the classroom. A summative evaluation is difficult and expensive because the goal of this educational software is not cognitive but procedural. Students are from many airports all over the world and the ultimate test can only be done at the turmac with Fokker 100 aircraft. The built-in tests are sufficient to guarantee educational quality in this case.

The research questions will now answered for this project referring to the analysis in the sections above. The implications of the answers are discussed briefly. See table 4.2 for the relationships between the questions and the aspects of the framework of Seligmann, Wijers and Sol (1989).

Question 1: How is the emphasis on instructional versus information technology aspects divided?
The "problem owner" of this project was Fokker's Maintenance Training department. An excellent blend of expertise in subject matter and teaching with visual media was available at this department. This guaranteed a strong emphasis on instruction. The management of the department MT themselves felt that they had to obtain expertise on professional project management, of large projects, educational technology, information technology and educational software development by an external party. BSO/Origin provided all of these sufficiently, thus a balanced emphasis on both instruction and information technology was possible under the final responsibility of the problem owner. The department MT did not call for support from other internal departments of Fokker except for buying the hardware and the network needed for the project. One representative of the automation department (Business System Unit) of the Product Support division, see figure 6.1, participated in the Steering Committee. Fokker provided expertise on subject matter, on integration with classroom teaching and it provided Fokker experts to monitor acceptance procedures carefully. BSO/Origin provided expertise on information technology, developing multimedia educational software and management of large projects. The balance of instructional versus information technology aspects was implemented explicitly in the contract between the partners. One of the results of this balanced emphasis was that the CBT group of Fokker was explicitly, as part of the contract, trained by BSO/Origin how to maintain the product, including software, visuals and audio segments. A manual was included on maintaining the educational software

produced. The costs of this "extras" were considered worthwhile by Fokker, see 6.2. Using this way of thinking, the small CBT group of Fokker proved to be able to revise the educational software for other aircraft and for other airlines.

Question 2: Is the product under development considered as a tool and/or as a medium?
This project shows strong emphasis on both instruction and media design. The simulations including the audiovisuals are highly interactive and attractive. The style of the written language and the voices used for the spoken language are tuned to the target group of international maintenance technicians. The choice to provide spoken word for all kinds for individual feedback and the attractive use of layout and colors on screen demonstrates that educational software was considered not only as a tool for training but also as a medium to engage the student. The simulations of many maintenance procedures with devices and equipment from the aircraft were realized in a highly interactive way using graphics and video. The interactivity could be improved further according to the final audit by BSO/Origin.

Educational software is considered to be a tool to improve the traditional courses fundamentally and as a medium to engage students from different cultures in many countries, in Europe, America and Asia, where aircraft are used and have to be maintained. This educational software package was not made as an isolated product for an open market but as an integrated and necessary component of the courses given at the Fokker MT centre or at the maintenance centres of airlines who have bought the Fokker 100. In international competition between aircraft manufacturers the use of modern media in maintenance training may be a small but important aspect of marketing policy. The media approach had an important position in the way of thinking for this project.

Question 3: Are the macro, meso, and micro levels of modelling distinguished explicitly?
All steps mentioned in chapter 4 can be recognized in the design process. The general macro and meso steps were made during the feasibility study and the pre-production phase, figure 6.9. At the micro level the semantic, syntactic and lexical steps were also made for procedures 1-4 during the pre-production phase. After this phase the other procedures were designed from the meso steps on, specifically for each procedure, but using the micro models made already during the pre-production phase. These approved models were used as templates and schemes to specify and fill in the components of the other 47 maintenance procedures. Only during the pre-production phase were models at all levels made. To save time, only exceptional details were modelled again for the next procedures. The implication of focussing on modelling at the levels macro, meso, and micro early in the project, during the pre-production phase, was, that the development of many maintenance procedures by experts from different disciplines in parallel could be organized efficiently.

Question 4: To what degree are different responsibilities within the development team defined and implemented?
As this was an out-house project it was carried out under contract, and because the MT department had very limited capacity for this project the project management and the responsibilities were prepared carefully. The responsibility for quality assurance (QA) for the project was assigned to BSO/Origin. In the quality assurance plan, made following HOEP, many necessary activities were listed and detailed for all phases of the project. Hundreds of activities were described including technical activities e.g. studio recordings for audio materials, digitizing these recordings and videodisc production. At the end of the project these lists of activities were refined into a manual for future use by Fokker during maintenance of the

educational software. Responsibility for each activity was assigned to one member of the team, a different member for carrying out the same activity for a different procedure if necessary. Other persons were responsible for the same activity in different procedures. With the same conscientiousness the tasks and responsibilities of all individual team members were defined beforehand as well as the tasks of all the groups, the form of communications and the agenda and procedure of meetings. The management of the MT department supervised the Steering Committee and the whole project.

It is clear that for a big project with two partners responsibilities should be defined clearly. The QA-plan used with this project was necessary and sufficient. The way of working was based on accurate definition of responsibilities beforehand and on continuous formal adjustment during the project. One of the implications of explicit changes of responsibilities during the project was that available expertise and man hours, 30 man years in total, could be used efficiently for development by parallel multi-disciplinary subteams.

Question 5: To what degree is communication realized explicitly between different disciplines? Supervision and control of communication between disciplines was the responsibility of the BSO/Origin project leader. The following experts were listed in the QA-plan: project manager, subject expert, educational specificator, programmer, junior programmer, AV designer, graphic designer, secretary. Many of the tasks of these experts were described in the QA-plan, the plan also described which communicating tasks were important. It appeared from an internal evaluation during the project that the communication between subject matter experts and specificators (educational technologists) was insufficient at the beginning of the project but that it improved strongly during the course of the project. Both disciplines learned a lot from each other yet not enough. Communication is difficult without experience in co-operation between disciplines, but one can learn it by practice in a structured project. It appears from a study made by Schouw (1992) on this project that communication between educational specificators and programmers was better than that between subject matter experts and specificators. The specificators and programmers, employees of the same company, were used to working at BSO/Origin as members of one team. As can be expected, experience in professional communication plays an important role. The second pair, subject matter experts and specificators, were not employees of the same company, moreover, the subject matter experts had a line function at Fokker and were, therefore, not "equal" members of the project team.

Communication is supported explicitly according to the guidelines in the QA-plan. Nevertheless communication for transference of thought proved to be difficult. Especially communication between subject matter expert and educational designer generated problems. Educational designers were members of a project team devoted to one or a few projects at a time. They became experienced in communicating within the team. The subject matter experts, however, had a line function in regular work, teaching in this project. They were involved in the educational software project on an ad hoc base. In this project the QA-plan facilitated good communication between all the disciplines except the subject matter experts. How to elicit knowledge about the subject matter, even if it is described extensively, is an underestimated problem. An implication of this experience is that communication between subject matter expert and specificator is very critical and can not be overestimated. To overcome this a solution may be to have at least one specificator in the team who is also an expert in the relevant subject matter. In most subject matter areas such person is as yet, difficult to find. Such a person will improve communication and probably the quality of the design, and he will contribute to enhance the efficiency of the design process by saving communication and meeting time. As reported by Schouw (1992) most specification problems were caused by problems with contents, see table 6.2.

Question 6: To what degree are the design and realization of the product separated into teams with different skills?
As the HOEP method was applied for the phasing of the development process separation of design and realization was a basic principle. In this project another separation can be seen, between subject matter specification and educational design. Separation has both advantages and disadvantages. Here separation was aimed at and at the same time conditions were created to compensate for the disadvantages of isolated work by explicitly planned communication between different teams.

A third separation was characteristic for this project. As more than 50 procedures had to be developed according to the same instructional structure, designing this structure in the pre-production phase was a separate process. This structure was modelled, see answer to research question 3 above, designed and realized in an iterative way while developing the first 4 procedures. When these procedures were realized the design of the structure, consisting of generic diagrams, many templates and textual descriptions was ready and by using this structure for the other procedures, designing became easier and faster.

The separation of design and realization was carried out consequently, this was considered to be a positive point. At the end of the project separation of the work was abolished to improve the efficient use of available man hours; however, it was perceived that, at the end of the project, responsibilities vanished and the quality of work decreased, Schouw (1992). An implication of this phenomena is that separation of design and realization can improve the quality of the final product; but this advantage must be paid for by spending more man power in total.

Question 7: Is the project directed by one person who has a clear image in mind of the characteristics of the final product?
The project leader of Fokker played the role of director of the development. He was, controlled and supported by the Steering Committee and had final responsibility for the quality of the product. The project leader had a background in aircraft engineering and has worked at different divisions of Fokker. He had a strong interest in and experience with visual forms and photography, but not at a professional level. He reviewed all products at the end of the numerous iterations. The project leader of the BSO/Origin team was not responsible for the contents; his role can be seen as the assistant director in charge of controlling internal communication between disciplines and partners.

As this project was large and complicated by the co-operation of two partners, the role of the Fokker project leader as a director was critical. He managed the project strongly, e.g. by requiring that all the staff of the external partner should work at the Fokker MT Centre to guarantee optimal communication. Although the Fokker project leader was not a formal "subject matter expert" he knew the application domain of aircraft engineering and maintenance well. For this reason he had the right profile to play the role of the director who has a clear image in mind of the characteristics of the final product. After finishing this project the project leader started directing the maintenance program for the educational software product by the Fokker CBT group. The role and the personality of the director is critical for the way of controlling. Two implications of this finding are that the project leader should be well acquainted with the application domain and that he should be conscious that their role is critical for the success of an educational software project.

Question 8: To what degree are explicit quality assurance measures provided?
Although the Fokker management (2 persons) had final responsibility the Steering Committee, meeting every 6-8 weeks, and the two project leaders, one each from Fokker and BSO/Origin, realized the daily management functions. Both project leaders had different and complementary tasks with an emphasis on contents and acceptance, the Fokker project leader, and production, the BSO/Origin project leader. The BSO/Origin project leader was responsible for quality control and updating the QA-plan. The development method was based on HOEP (section 2.1.7). During all phases the management was very strong but flexible when required by changing conditions. The management regime was adjusted many times when conditions were changed and new problems came up. When capacity problems occurred tasks and responsibilities were changed, sometimes after a period of short training to achieve new skills. Management had problems with the availability of subject matter experts, being instructors in charge off teaching maintenance training in the traditional way.

Among others, the head of the Fokker MT department was the general manager of the whole project. He had a vision of the aims and the required quality of the final product and he also demonstrated leadership, at a distance, to guide the project. He was responsible for evaluating the progress of the designs and the products under development using the norms in aircraft industry to strive for the highest quality. The Fokker project leader behaved as the real director who not only controlled details of the project management but, first of all, the quality of the product. He was aware of the way the content and the media use could influence the usability of educational software. The BSO/Origin project leader had a lot of experience in managing educational software projects. He was experienced in using the HOEP method, including the software engineering aspects. The careful design of the user interface separating semantic, syntactic and lexical aspects was one of the results. The video components and the audio feedback are also designed carefully. The most difficult aspect of the design activity was to specify the subject matter explicitly and completely. Many errors discovered during the last phase of the project were due to small faults made while specifying the subject matter.

The development method used is similar to the way of working described in chapter 4. Although the macro, meso, micro levels were not applied explicitly they can be distinguished implicitly. The macro level is reported in 6.2.1 and 6.2.2. The meso level is reported in 6.2.3 and 6.4.1-4. The micro level is reported in 6.4.4. The QA-plan was worked out for this project initially and it was revised many times during the project, as this was the assignment for one partner in the project. The QA-plan itself was not so important but the fact that it was applied in the generic form and tuned to the context of the project was important.

7. CASE 3: LEARNING A SECOND LANGUAGE

This case study was carried out in primary education as part of a national program to develop, in a professional way, a considerable amount of educational software to be delivered through educational publishers. The subject matter of the project chosen was learning/exercising Dutch as a second language. There is a special need for this subject because of the increasing numbers of refugees and immigrants. This case was selected for the following reasons. First, the topic of learning a second language is not new and much insight is available about the contents and the didactic strategies possible and suitable for different target groups and ages. Second, this was a multi-partner project, each partner, the national government providing funding, the program development manager, and the educational publisher, having a different "problem" to be solved. Third, the product belongs to an accepted traditional method for teaching reading at primary level in use by tens of thousands of children. Fourth, use of the product depends on the educational market. Teachers can use the product for all children and/or for remedial teaching. Fifth, the educational software produced provided more than 10 hours of session time in total and was integrated with other media.

First the context of the program and the project is described in general terms in 7.1. Then the characteristics of the educational product are given in 7.2. All available internal project documents were studied and the two project leaders of the developing team and the publisher were interviewed. The development process is analyzed in sections 7.3 - 7.6 using the framework presented in chapter 2. Finally, the research questions are answered in 7.7.

7.1 The POCO program

Following on from the Dutch national stimulation program to introduce information technology into education (INSP from 1984-1988, see section 3.1.1) the Dutch government decided to continue to stimulate the use of information technology in education and several new, large programs were started at the end of the INSP for this purpose, e.g. the PRINT project, see section 2.2.8. One of the outcomes of the INSP was that insufficient amounts of comprehensive educational software were produced and available on the educational market at the time. So, among others, one of the important aims was to develop a significant amount of educational software. To achieve this the POCO program was established. POCO stands for Software Development for Computers in Education (Programmatuur Ontwikkeling voor Computers in het Onderwijs). The POCO program was carried out during 1988-1992. The aim was to produce enough educational software to support 5% of the lessons in the curriculum of secondary (general and vocational) education and 1% of the lessons in the curriculum of primary education. The total budget was, at the time, DFL. 25 million. The POCO program officially ended at the end 1992 and delivered 55 products containing more than 1000 hours of educational software. About 70-80 net[1] man year were needed for these productions, generated by about 350 persons (POCO, 1992).

1. Priority and commercialising processes are not taken into account. These contributed about 15 man years to the whole POCO program. With this work included the development ratio is about 150:1.

The development ratio[1] for the whole program was about 120:1.

The POCO program was initiated and funded by the Dutch government, parliament approved the goals and the budget. The program management was put out to contract with Educational Computing Consortium[2] (ECC). ECC is a private enterprise founded by skilled staff who had worked on the INSP program.

Program management was directed by the Minister of Education and Science to initiate, perform and control each of the following processes:

1. Deciding on priorities for subject areas related to curricula. The procedure was the following: the POCO management had to describe possible priorities in "whitebooks" based on consultation with curriculum experts and educational specialists from different educational domains and levels. These whitebooks had to be submitted to the minister, who would decide, assisted by his own independent advisers, which products should be realized.
2. Producing functional specifications of products for educational software. A team had to be formed for every product consisting of at least one experienced subject matter specialist, an educational technologist and a person who was able to produce prototypes. Specifications of the software, specifications of supporting data, written materials and criteria for field tests were required.
3. Organizing tender procedures for the realization of the designs by the software house that submitted the best offer. Realization of the designs should be done by one or more subcontractors. This ment design and realization had to be separated almost completely. The POCO management had to perform an acceptance procedure on the products delivered. A field test would be part of the acceptance procedure in most cases.
4. These delivered products should be transformed into products that could be distributed on the educational market at their own risk by educational publishers. The commercial channel of the educational publishers must be used to sell and distribute products and, in the case of generic programs, the educational contents belonging to the generic program.

The POCO organization is depicted in figure 7.1 The POCO program represents a way of producing educational software for the open market for formal education materials. With respect to educational software this market can be considered as semi-commercial. Semi-commercial means that schools and consumers are free to buy a product or not, but the price cannot cover all the development costs. This is necessary due to the small size of the Dutch educational software market. About 20 million native Dutch speakers live in the Netherlands and Belgium. More than 10,000[3] schools exist for this population including about 9,000 primary schools. One cannot expect to sell more than a few hundred[4] licenses for an average educational software product on this market. Therefore, in the POCO program development costs upto the final pre-distribution products were payed for by the Ministry of Education and Science. The pre-distribution product was given free, but with well-described conditions to one or more educational publishers. The marketing, distribution and the production processes of the media, books, audio tapes and educational software, were performed commercially and these costs have to be taken into the cost benefit analysis of a published product. Maintenance of a product also has to be done by the publishers on a commercial basis. POCO management choose for an "industrial" approach to the development process of educational software.

1. The development ratio is defined as the hours of development time necessary for one hour of average session time.
2. In 1992 ECC was taken over by Andersen Consulting to form Andersen Consulting-ECC.
3. This is the number for the Netherlands.
4. This is the opinion of the author.

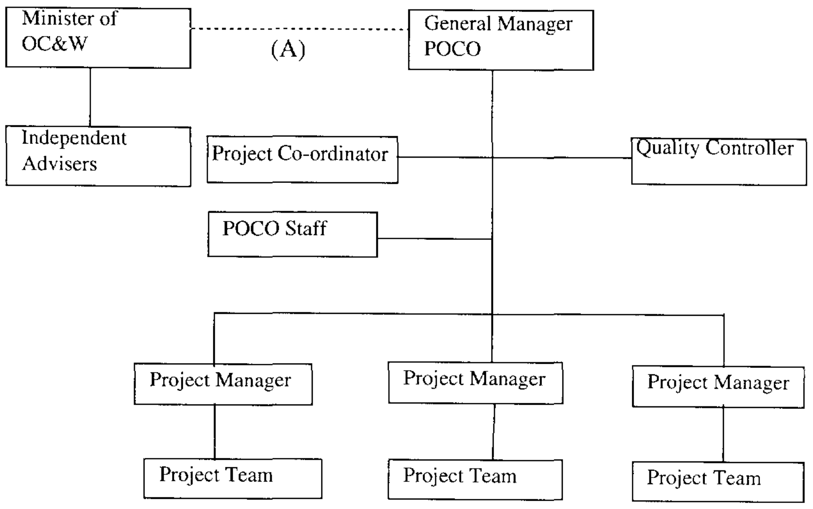


Figure 7.1 POCO organisation. The formal deliberation (A) supports the semi-annual formal progress reports of the POCO Manager and the reaction of the Ministry. As for the status of the QA-plan and the POCO products this formal deliberation is functioning as Change Control Board (POCO, 1991).

The products developed by the POCO program were not developed for specific "problem-owners" (see 1.3) but for schools and students in general. This introduces the problem of how to know the market, i.e. to know characteristics of the partners in the market: the educational publishers with their strategies and policies, and at the same time the characteristics of the "consumers" of the product. The schools[1] that might buy the products, the teachers interested in using and advising on buying the products and last, but not least, the students having to use the products interactively were considered to be the consumers. Therefore the POCO program represents a special case for developing educational software. The Ministry of Education was the holder of a mandate given by parliament, but not the problem owner at the level of product development. The Ministry commissioned the POCO program and demanded the general constraints for managing the POCO program mentioned above in points 1-4. These constraints mandate professional management and a formal QA-plan to control the progress of such a large program of projects on different subject matters and for different target groups.

ECC had to manage the development without expertise in all subject matters and without having the capacity for the realization of all the projects. ECC had to implement a temporary operational infrastructure that could be referred to as an *educational software factory*. This is a rare situation that is completely different from the in-house development described in chapter 5 and the out-house development described in chapter 6. Given this difference it is, however, interesting to study the development of one product under these conditions. The project chosen for this case study can also be considered as part of a series of consumer oriented projects. All the products of the POCO program were developed for the educational market in the

1. It is not assumed that students will buy educational software for personal use. Schools will buy the products.

Netherlands and Belgium.

To cope with the management of this program for the development of so many educational software products it was decided to organize the development into two cycles, each taking about two years of development time.

During the first cycle 16 products were developed and published, consisting of:
2 for primary education (language and arithmetic)
6 for secondary education (languages, geography, mathematics, etc.)
8 for vocational training (healthcare, mechanics, technical drawing, clothing design, etc.).

During the second cycle 39 products were developed and published, consisting of:
8 for primary education (reading, ciphering, geography, etc.);
13 for secondary education (languages, mathematics, physics, sociology, etc.);
18 for vocational training (administration, budgeting, logistics, datacom, physics, mechanical constructions, etc.).

The products were sold for 100 - 3000 DFL[1] per site license or about 25 DFL per student license.

During the second cycle more emphasis was put on market analysis (POCO, 1992). Several strategies were developed to estimate the interest in a product. The following approach was chosen. An idea for development of a product was worked out first as a product definition. This product definition and a list of relevant questions were sent to five (this is an average) experienced users of educational software or other subject matter experts on relevant topics. The answers of these users resulted in a first impression of the demand for the product. Then a real market analysis was done of possible future users of the product. This analysis was performed in different ways depending on the product, the characteristics of the target groups and schools. The primary goal was to get a go/nogo decision for the product.

Another instrument in marketing was to sell not only site licenses but also student licenses, the latter to stimulate demand driven selling and costs depending on real use by students.

In general the co-operation with the educational publishers on the contents of the educational software went well. The formal contacts concerning acceptance procedures and royalties were, however, confusing and difficult during the first cycle. Some carefully formulated standard contracts for the publishers were made for the second cycle. Although these better contracts improved matters a lot, during the second cycle problems arose with some of the publishers. Probably these were caused by low expectations of the market on the part of the educational publishers (POCO, 1992).

During the first cycle it turned out that the software houses who realized the designs were working too technically. Total separation between design and realization introduced communication problems. During the second cycle, for some projects, an employee of the software house who was interested in the later realization of the design was assigned part time to the specification team to contribute to the functional design. A better understanding of the functionality of the design within the realization team was the result. A disadvantage of this solution was that the offering phase became less clear, less open and less objective (POCO, 1992).

The POCO program ended without any excess of budgets but with a one year excess of time. This was caused by several reasons (POCO, 1992):

1. The price of most products was about 500 DFL. Some general and powerful products for secondary vocational education (MBO) had higher prices.

1. In the initial planning 3 weeks were scheduled for market analysis as part of the feasibility study. It turned out to be 10 weeks or more for many projects;
2. The availability of senior subject matter experts was a big problem for the planning. Specification started late for some projects, sometimes 6 months late[1];
3. The publishers had other priorities for their authors and for policies. The commercialization of a developed product was often more than 6 months late.

The general management of the total POCO program of all projects took about 8% of the budget.

The educational software for Dutch as a second language was developed during the second cycle (1990-1992). At the time the "educational software factory" of POCO was working at an optimal level using a second improved version of a quality assurance plan based on the IEEE 730 standard. The project described here is a relatively small one compared to the cases in chapters 5 and 6, but it was supervised by a team managing more than 10 such projects simultaneously using the same methods and tools. The POCO program as a whole is considerably larger than the projects described in chapters 5 and 6.

7.2 The product Dutch as a second language *WOORDWIJS*

The idea to develop educational software for Dutch as a second language was launched during a meeting of POCO management with educational publishers on the market prospects of products to be developed during the second POCO cycle. Shortly after the first idea the Ministry of Education requested an investigation of the feasibility of this subject explicitly, because it was not included in the priority list of subjects registered, discussed and approved for the second cycle. The POCO management added this subject to the list and produced the feasibility study at short notice.

During the first cycle of the POCO program the difficulties of educating most ethnic minorities were not yet (in 1988) on the Dutch political agenda. This changed drastically with the arrival of new refugees. Many sectors of society including schools, educational publishers and the Ministry of Education and Science became conscious of the necessity to improve the education of all ethnic minorities. Hence the subject of educational software for Dutch as a second language was inserted into the priority list later.

7.2.1 WHY EDUCATIONAL SOFTWARE?

About 5% of the Dutch population come from ethnic minorities. This number is increasing. In some cities about 30% of the children of primary school age belong to minorities. At some schools more than 90% of the children are from minorities. The large difference in reading vocabularies of Dutch words between native Dutch speaking children and children from these minorities is striking. Reading vocabulary is very important for the further education of children (Verhoeven and Vermeer, 1985; Coenen and Vermeer, 1988; Theunissen, 1990). Children with poor reading vocabularies produce significantly lower results in school. Mastery of vocabulary differs dramatically between ethnic minority children and native Dutch speaking children. The gap is measured to be 4 years for children[2] of Turkish families in the Netherlands aged 8 compared to native Dutch children of age 4. Vocabulary can be improved significantly

1. Senior subject matter experts were mostly involved in important line functions in the field they could not leave at short notice.
2. First generation children with parents not speaking fluently Dutch.

with computer based exercises. Children should be able to read more texts within different semantic contexts, to do individual exercises when required and performance should be measured by computer based tests. The results of the test need to be registered for teachers to decide on which exercise and when an exercise should be followed individually. If these exercises fit existing second language methods then teachers can co-ordinate exercises on a daily basis without much investment of time.

The computer based exercises were required to place particular emphasize on consolidating words in memory and on testing for the degree of mastery. It is well known from literature (Appel, 1989) and experience that it is important to decide *which words* have to be learned, in which *sequence* they have to be learned and *how* they are presented and learned. "How" is important to motivate the children and to offer rich semantic variability.

A feasibility study was carried out for the Ministery of Education. Then during the product definition, like other POCO projects, a review of a large number of second language methods was carried out. The didactic goals and objectives were derived and specified from this study. The roles of students, teachers, subject materials and software functions were also specified in detail. After the feasibility study the POCO management looked for interested educational publishers. First agreements were made to base the aimed educational software product on the method for continuing reading *Wie dit leest* for the grade levels 4-8 (Aarnoutse and Van de Wouw, 1990) produced by Uitgeverij Zwijsen b.v. at Tilburg. This method uses a list of more than 3000 Dutch words. This new reading method consists of a comprehensive teacher's manual and many illustrated reading books, exercise books and audio tapes. The market for educational software for Dutch as a second language based on this method was estimated to be about 400 primary schools in the Netherlands (30-40% of method users).

7.2.2 REQUIREMENTS

The scientific literature on the subject was studied thoroughly during the product definition. Many alternative types of exercises were described and prototyped on paper. A high level of didactic expertise and educational experience on the subject of second language learning were available within the design team. The following general requirements were specified for the educational software to be developed:

1. the product should fit into well known methods for Dutch as a second language;
2. the target group of children: primary education levels 4-8 (8-12 years old). Children from different ethnic minority groups should benefit from the product regardless of their native language;
3. the user interface of the educational software should be suited to the cognitive and emotional level of the children;
4. children should be able to use the educational software independently, alone or in pairs after a short introduction by the teacher;
5. the product should be focussed on exercising Dutch vocabulary after an explanation and "semantisizing" of new words in the classroom;
6. as much as possible, different types of exercises should be included to offer a motivating variety of exercises over all levels.
7. teachers should not need too much time to control the use of the program by children;
8. registration of the results of children should be complete and easily accessible to teachers;
9. an authoring tool should be realized for the publishers to fill in the exercises;
10. the educational software should run under MS Windows on the Comenius computers[1] used in Dutch schools.

7.2.3 THE PRODUCT

The *Woordwijs* product is marketed as educational software for children of ethnic minorities and other children with a poor vocabulary in the grade levels 4 to 8 of the Dutch primary school system. The content of the product is organized in one module for each level (5 modules in total). Each module consists of 10 themes. Each theme consists of a different number of exercises of many types. Within the context of this POCO project the module for grade level 4 was completely developed and delivered. Only the development of this module is described here. The other modules were filled in by the publisher after the project using the same authoring editors for all types of exercises.

The product *Woordwijs* is based on the new Dutch reading method *Wie dit leest.* This method consists of 150 lessons of 40 minutes for grade level 4, see table 7.1.

Subject	Number of lessons
technical reading	50
understanding reading	30
information acquisition	10
stimulation of reading	30
free reading (no texts and exercises)	30

Table 7.1: Lesson scheduling for grade level 4 of the method *Wie dit leest.*

This module consists of 10 units on different themes: environment, nature, travelling, emotion, stories, living together, culture, past and future, adventures, and language. 12 lessons are specified on each theme in the teacher's manual. The free reading of many books for children is suggested for the lessons. The teachers must make their own choices. The module for grade level 4 consists of 90 different exercises.

Eleven types of exercises are provided:

1. Matching words (1) and sentences (2). At the top of the screen four pictures, words or sentences are presented, at the bottom 4 other words or sentences. The student has to click an "object" on the bottom and then click an "object" from the top to which the one below belongs. If the student thinks he is ready he clicks on the OK button. An example would be the paired lists tall/small, white/black, old/young and yes/no. Words with pictures and sentences with sentences can also be paired, see fig. 7.2a.
2. Duo words (3). The student reads a text on screen in which some words are marked by one color. Below the text a series of pairs of empty boxes are shown. The student can click on a word in the text and on a box to form a pair of words belonging together.
3. Selecting the wrong word in a list (4), see fig 7.2b.
4. Categorizing words (5) and sentences (6). The student has to categorize sequentially presented words, pictures or sentences into 1-3 categories, e.g. mammals, tools and countries or amusing and sad [style A]

1. In a separate project Dutch Ministry of Education and Science provided one Comenius computer per 60 children per primary school. These computers have standard features to run MS Windows and are delivered with some general educational software products especially made for this Comenius project. Many schools bought more Comenius computers from their own budget.

5. Categorizing words (7, 8 and 9) [style B], see fig. 7.2c.
6. Placing words in the right order (10), e.g. morning, afternoon and night. This exercise is illustrated with different types of animations, using metaphors of spatial and time sequence.
7. Filling in matrices of words (11) [style A]. Vertically categories are indicated, e.g. (bee, insect) and (duck, bird). Horizontally words are indicated that correspond to the vertical categories or not, e.g. swimming, buzzing, barking and flying. The student has to place crosses in the right cells of the matrix.
8. Filling in matrices of words (12) [style B].
9. Filling gaps in text bodies (13) [style A]. A text is presented. Questions about the text are presented sequentially with three possible answers. An answer is chosen by clicking on it.
10. Filling gaps in text bodies (14) [style B], see fig. 7.2d.
11. Multiple choice questions (15). A text is presented. Questions about the text are presented sequentially with possible answers. An answer is chosen by clicking on it.

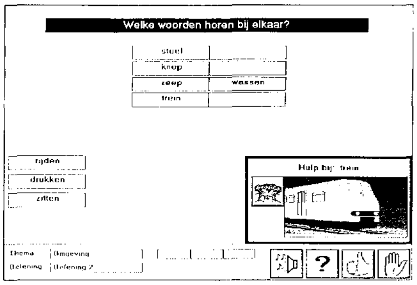
These 11 types of exercises were worked out into 15 different forms which can be recognized by the student. Each exercise can be done within a few minutes (3-10 minutes). After completing an exercise an attractive animation is showed for a few seconds. A series of circus animations are available for the grade level 4.

All types of exercises are implemented in different attractive forms to enhance the motivation and fun for the children, see figure 7.2 for examples of four types of exercises. Pictures[1] and animations are added in the modules for the lowest grade levels. When the computer has a standard audio card children can hear the pronunciation of words[2] presented on the screen. A picture can also be shown on request to the student to explain the meaning of the word selected currently.

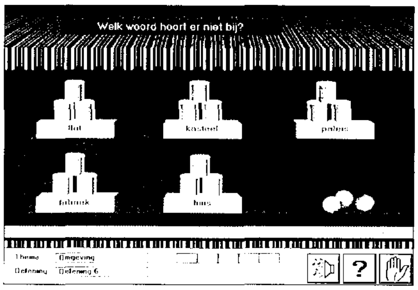
During all exercises help and voice output are available to students, if the teacher permits this. Two comprehensive tests are provided for each grade level to score the progress of the students. The tests are carried out using an adjusted form of the multiple choice type exercise.

An essential component of *Woordwijs* is the teachers module, see figure 7.3. The teacher has to administer the individual students and place them in groups. Only registered students can do the exercises. The teacher also has to indicate which exercises have to be done by specific students, individually or in pairs, with extra help and supporting speech output or not. All answers given by students are recorded during the session. The teacher can inspect which exercises have been completed by each student. The development of the teachers module is not discussed further because it has all the characteristics of conventional interactive information systems.

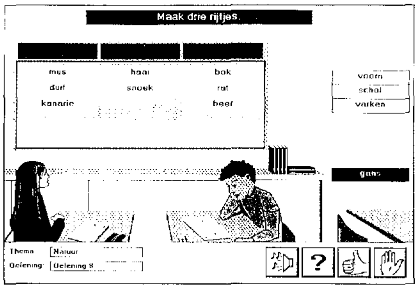
1. Pictures are only available for word having a "difficult" meaning for children.
2. Voice feedback is only available for words which are technically "difficult" to read.



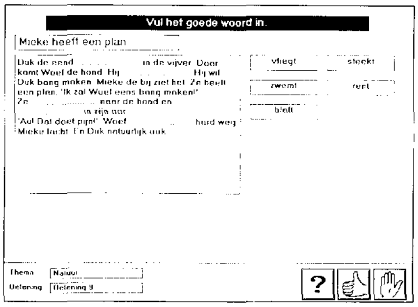
a



b



c

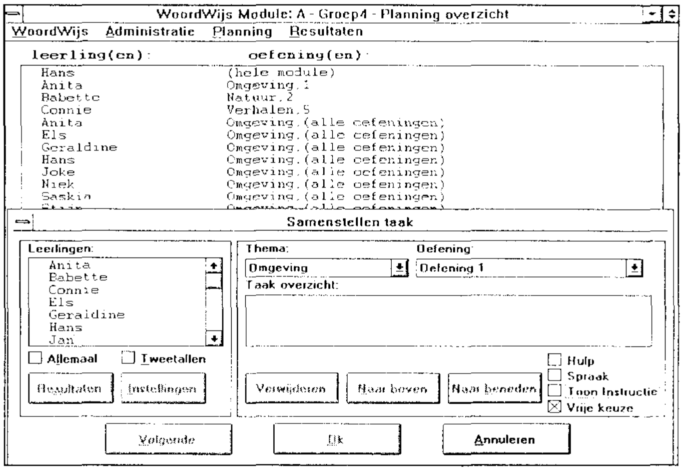


Mieke heeft een plan

Duk de eend .... in de vijver. Daar komt Woef de hond. Hij .... Hij wil Duk bang maken. Mieke de bij ziet het. Ze heeft een plan. 'Ik zal Woef eens bang maken!' Ze .... naar de hond en .... in zijn oor. 'Au! Dat doet pijn!' Woef .... hard weg. Mieke lacht. En Duk natuurlijk ook.

d

Figure 7.2 Example of four types of exercises of *Woordwijs*. a. matching words (1); b. selecting the wrong word in a list (4); c. categorizing words (8); d. filling gaps (13). The numbers refer to the types mentioned in the text. (Courtesy of Aarnoutse, vd Wouw: Woordwijs, Zwijsen, Tilburg, 1993).



leerling(en):	oefening(en):
Hans	(hele module)
Anita	Omgeving.1
Babette	Natuur.2
Connie	Verhalen.5
Anita	Omgeving.(alle oefeningen)
Els	Omgeving.(alle oefeningen)
Geraldine	Omgeving.(alle oefeningen)
Hans	Omgeving.(alle oefeningen)
Joke	Omgeving.(alle oefeningen)
Niek	Omgeving.(alle oefeningen)
Saskia	Omgeving.(alle oefeningen)

Samenstellen taak

Leerlingen: Anita, Babette, Connie, Els, Geraldine, Hans, Jan

Thema: Omgeving

Oefening: Oefening 1

Taak overzicht:

☐ Hulp
☐ Spraak
☐ Toon Instructie
☒ Vrije keuze

Figure 7.3 Example of one of the screens for the teacher to manage the use of *Woordwijs*. (Courtesy of Aarnoutse, vd Wouw: Woordwijs, Zwijsen, Tilburg, 1993).

Editors are provided to fill (author) of all types of exercises. Eight editors were developed for the 11 types of exercises. Some similar but different exercises can be filled with one editor using a special switch or option. The basic structure of all types of exercises was fixed and cannot be changed. The exercise editors have different complexities. All parameters of each type of exercise can be specified, e.g. name of the exercise, correct answers and feedback options, and the contents can be filled-in on the screen or linked to from external files, audio and picture files. The exercise editors were designed and programmed in parallel with the programming of the exercise drivers for the students. The exercise drivers are linked to the exercise editors. It is easy to inspect an exercise in student mode during the editing of parameters and contents. The editors were used for data entry of the contents of all exercises and to improve the contents after formal evaluation and try outs with students. These editors are not considered to be part of the educational software product for the market.

The product *Woordwijs* for grade level 4 consists of 90[1] exercises of 11 types and 2 general tests, 7 Mbytes of code and data, including speech output. Development including filling all exercises took about 2 men years from ECC (exclusive publisher Zwijsen). The publisher Zwijsen later designed and filled the contents for the grade levels 5-8 using the same authoring editors. This work is not taken into account in the project described here. The contents for all grade levels were completed in 1994.

7.3 Way of thinking

All educational software developed within the POCO program had to be marketed and sold by educational publishers. The selection process of the educational software to be developed was considered to be critical. Therefore selection was organized carefully according to a large scale plan for both cycles. Participation of educational technologists, subject matter experts, senior teachers, computer scientists, institutes for curriculum development and educational publishers was provided. Another critical factor was the availability of enough hardware and information technology skills on schools. Without a good infrastructure in the schools, educational software is worthless. This factor depended on other programs of the Ministry which provided hardware and were not influenced by the POCO program. It can be concluded from the careful attention paid to the selection process, that the way of thinking is characterized by a responsible approach to problem analysis in the educational field and to the market.

The necessity to develop for a market caused POCO to distinguish and separate the four relatively independent steps of development mentioned above: selection of subjects, specification of products, realization of products and delivery to publishers for marketing at their expense (Moonen and Schoenmaker, 1992). The implementation of the products in schools was completely out of the scope of the POCO program. It was at the time uncertain how the publisher would succeed in marketing the *Woordwijs* product, as was the case for all educational software for formal education in the Netherlands. The goal of the project was not only to deliver a product for grade level 4 to the publisher, but also to prepare and instruct the publisher in how to fill in the contents of and how to use the authoring editors for the modules for the other grade levels.

Despite considerable uncertainty concerning the market perspectives, the subject matter and characteristics of the target group were known accurately to the developers.

Due to the large scale of the POCO program it was possible to provide high professionalism in

1. Using an average session time of 6 minutes a total of 9 hours of educational software was produced for this project (for grade level 4).

management at the main level and at the project level. The method used was based upon experiences in the INSP program, the HOEP method, see section 2.2.7, enhanced with new insights into educational design and management. The first version of the POCO Reference Book (POCO, 1989) and the POCO Quality Assurance Plan (POCO, 1991) were the creed of the way of thinking.

The professionalism of the POCO management can best be characterized by the concept *educational software factory.* All 55 projects (of which *Woordwijs* is only one) were carried out using the same QA-plan consisting of chapters for among other things: management, documentation, methods and techniques, reviews and audits, configuration management, reporting problems and revisions, methods and techniques for quality assurance, financial planning, and user interface guidelines. This QA-plan was based on the IEEE 730 standards. In this plan organizational, educational, technical, subject matter and ergonomic aspects are considered. Roles and responsibilities of members of the teams and the different disciplines were also prescribed beforehand. A revised version of the QA-plan was made (POCO, 1991) for the second cycle of POCO.

7.4 Way of modelling

The method described in POCO QA-Plan version 2.0 is based on four models, see also Schoenmaker et al. (1990) and Schoenmaker (1993):
Object System Model (OSM);
Educational Design (ED);
Functional Design (FD), Informal Functional Design (IFD) and Formal Functional Design (FFD);
Technical Design (TD).

OSM and ED are models in terms of instruction and learning of the subject matter. These are informal models but well structured. FFD and TD are based on a formal software engineering method Yourdon-Real Time, see Hatley and Pirbhai (1988). These models were used to describe two different systems: the object system and the educational software system. The object system constitutes the educational environment of educational software. OSM was used to describe this object system. ED, FD and TD were used to describe the educational software, because of the small scale of the technical part of the project the FD and the TD were made at once and described in different sections of one report.

OSM aims to describe the role of the educational software in the educational context. It is the first design specification step. It will be referred to by the educational designers, for the ED, and by the information analist for the FD. The programmers did not refer to the OSM. The OSM also provides a medium for later communication with external partners, e.g. publishers, subject matter experts and teachers. According to the QA-Plan the OSM document had the following structure:

1. Introduction
2. Subject matters and learning objectives
2.1 Description of the target group
2.2 Description of the aimed knowledge and skills
2.3 Description of the prerequisites of the student
2.4 Description of the learning objectives in concrete terms
3. Educational context
3.1 Didactics of the subject matter
3.2 Didactic steps, eliciting interest, orientating, testing and teaching pre-knowledge,

presenting and explaining subject matter, applying and exercising subject matter, evaluating and deciding, reteaching and knowledge transfer
3.3 Instruction strategy
3.4 Subject matter
3.5 Task analysis
3.6 Organisational aspects (optional: organisation of teaching in groups and schedules)
4. Roles of the actors: student, teacher, educational software, author, other media.
4.1 Roles of level 1: teaching/learning
4.2 Roles of level 2: preparation and evaluation of lesson
4.3 Roles of level 3: preparation of generative lesson

The OSM model was applied accurately in this project. It provided the basic model to be worked out during the next steps of the specification phase.

ED consists of details of the OSM, especially components and tasks, description of activities to be performed by the educational software. The structure of the subject matter was modelled in the ED using Entity Relation Diagrams (ERD's) and Object Modelling Technique (OMT). Task modelling was described with decomposition diagrams and with text.

It was natural to distinguish structure and content due to the *Woordwijs* product's pure drill and practise nature. A small set of 15 types of exercises was designed based on the method *Wie dit leest*. This method provided the semantic and didactic framework for the contents of the exercises. The first design problem focussed on the concepts of the exercises and their syntactic and lexical properties. The second design problem was to choose suitable contents belonging to the existing themes described in the written student material of the method. The general model of the relationship between exercises and contents was depicted using ERD, taken from the functional design (figure 7.4). The entity exercise is considered in the dataflow diagram of figure 7.5. All 15 types of exercises were specified in the ED in a standard scenario described in plain text and some figures:

1. presentation and interaction (syntactic and lexical design);
2. feedback for right/wrong answers (syntactic and lexical design);
3. how the student can ask for help;
4. example of the screen with sensible content;
5. specification of speech output;
6. way of registration of student results;
7. how the teacher can adjust the exercises (not the contents);
8. requirements for the authoring editor for the type of exercises.

The scenarios for the standard tests were made. These tests provide the teacher with a means to track the progress of the student.

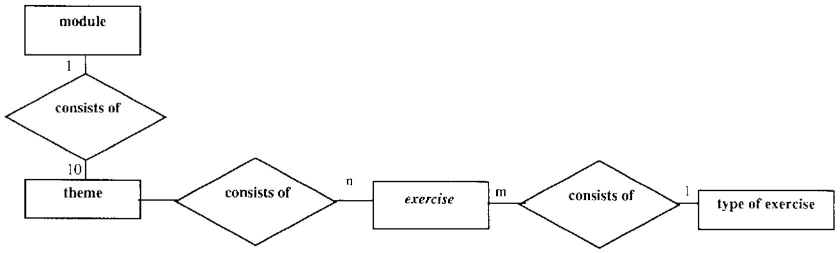


Figure 7.4 Entity Relation Diagram describing the model for the structure of the exercises. A module consists always of 10 themes.

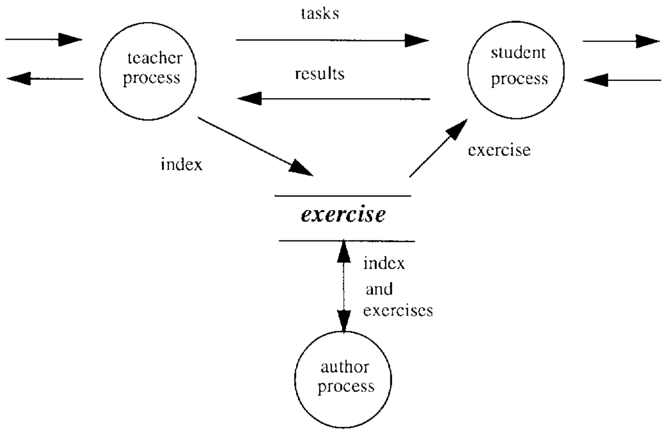


Figure 7.5 Data Flow Model of the entity *exercise* (see figure 7.4) at the highest level

7.5 Way of working

All projects within the POCO program were carried out according to the general POCO QA-plan (POCO, 1991). Among other things, this QA-plan prescribes, in small detail, the way of working including the assignments and responsibilities of all disciplines and persons involved in a project.

A Steering Committee (SC) was formed for this project. The Members of the SC were the general POCO manager, the project co-ordinator, the manager of the project later augmented by a general manager from the publisher and the project leader of the *Wie dit leest* method for reading. The principle of separating design and realization was maintained. Another fortunate issue was that the project manager of the team had considerable expertise in project management for developing educational software; she was also a subject matter expert on Dutch as a second language and an experienced educational technologist.

The POCO QA-plan prescribed the development method. In this method educational software development is considered to be a branch of software engineering with special emphasis on interaction and the use of multimedia data (Schoenmaker et al., 1990). The phases and goals are described in table 7.2.

A mini QA-plan was made for each project during the start up phase. This was derived from the QA-plan and consisted of a subset of the guidelines considered to be applicable to the project. The mini QA-plan could be seen to be tuned to the context and size of the project and referring to the general POCO QA-plan.

Phase	Goal	Actors[a]
(0. priority fixing)	outside the scope of the case study	-
1. start-up	identification of an idea, finding SME's, writing a project plan	(1) with internal and external consultants
2. product definition	further description in educational context	(1,2,3,4)
3. feasibility study	criteria: market prospective, technology, organisational and educational aspects, interest from publishers, functionality, budget, relation with other POCO projects. A NOGO decision is possible.	(1,4,5,6,7)
4. product specification	Object System Model, Educational Design, Functional Design, Technical Design	(1,2,3,4,5)
5. make an offer/quotation	to put out to contract the software component to a (external) software house	(1,2,5,8,9)
6. realisation	realisation of all components including manuals and exercise books	(1,2,4,5,6,10,11,12,13)
7. acceptance	acceptance of all components and their integration	(1,2,3,4,5,6,11,12,13)
8. field test	testing in the classroom with students from target group	(1,4,7,14)
9. revision	revision of all components	(1,2,4,5,10,11,12)
(10. preparing for the market)	outside the scope of the case study	(1,6,11,12,15,16)
(11. distribution)	outside the scope of the case study	(6)

Table 7.2: Phases and goals of the way of working (QA-plan version 2.0)

a. Actors are the following:

(1) Project leader	**(9) Employee of software house**
(2) Educational technologist	**(10) Programmer**
(3) Subject matter expert	**(11) Author**
(4) Evaluator	**(12) Graphic designer**
(5) Information engineer	**(13) Data provider**
(6) Publisher	**(14) Student as user**
(7) Teacher as user	**(15) Editor of a publisher**
(8) General POCO manager	**(16) Software supplier**

The required management reports, the products and the quality assurance actions are described for each phase in table 7.2. The role of the participating actors is also described. This project was performed according to this schedule with some minor modifications.

The start-up phase was performed according to the guidelines in the QA-plan. The product idea was identified, and the project manager functioned as the manager/SME throughout the feasibility study during the start-up phase. The conclusions of this study were positive. Product specifications were made simultaneously for the structure of the exercises, the drivers and the editors.

The design of the 15 types of exercises was done by the project team. The drivers and the editors were designed and later released with some illustrative contents. Paper and software prototyping was carried out. The editors were designed to co-operate with the drivers to support fast testing in student mode during editing.

The functional design FD and the technical design TD were described together in one document during product specification. The functional design at the level of the exercises was described in terms of the exercise drivers, partly in pseudocode. The conceptual UI design was done first by prototyping on paper by the subject matter expert and later using Visual Basic by the same subject matter expert. The editor (human) of the method from the publisher, who edited the textbooks of the method, made the syntactic and lexical designs later.

The offering phase was very fast because the realization was planned to be in-house at the ECC. Realization of the designs had to be done by software houses after public tender procedures, however, for the *Woordwijs* project, realization was done by an internal programming team of the ECC. This and all other products for primary education were programmed by ECC internally for the following reasons:
1. Experience with the realization of earlier projects for secondary education and vocational education showed that the contracted software houses yielded poor results with respect to user interface and performance at the time (1990). These aspects were considered to be crucial for primary level education.
2. Software houses in the Netherlands had little expertise at the time in programming interactive software for MS Windows, especially for the Comenius computer.
3. The programming department of ECC proved to have more expertise on this area and it was interested to invest in further expertise, especially in Visual Basic. The software houses had not decided to invest in using such tools, at this stage.

Despite the fact that the realization was carried out within ECC by another department the separation between design and realization was implemented formally according the QA-plan. Of course, informal communication about functional aspects, with people who were involved during the design phase, was possible during the realization phase. The realization phase was done by two people from ECC, one of them also participated in the functional design.

The filling of the exercises with the contents of the teaching method *Wie dit leest* was a second project within the *Woordwijs* project. This is due to the twofold technical goal of the project. All exercises and the software to support them had to be developed and the contents of more than thousand exercises, for all grade levels, had to be chosen and released based on the vocabulary used within the method *Wie dit leest*. It was decided and agreed by ECC and the publisher that ECC was responsible for realizing the drivers and editors and that the publisher was responsible for filling in all contents, words, sentences, pictures and voice recordings. The editors were designed to be used by the authors of the teaching method of the publisher. The chief author and the assistant authors of *Wie dit leest* however decided to fill in the contents of all exercises, 90 for grade level 4 on paper. Forms were made for this purpose. The review of the contents filled in for the exercises was done on paper, and also the following revisions. After completion and internal acceptance of the many series of exercises on paper other people from the publisher used the exercise editors only as data entry tools.

The field test was done by an internal department of ECC that has a lot of experience with field tests of educational software. The result of this field test was positive and some minor improvements were advised. The same department published, after this project, a consumer test of *Woordwijs* (SCEN, 1992).

The revisions after the acceptance procedure were done internally by the publisher, the few technical software changes were performed by ECC.

Preparing for the market is outside the scope of this case study, this was done independently by the publisher.

7.6 Way of controlling

Normally the problem owner looking for a solution to his problem plays an indirect but important role in controlling the outcome of a project. With this project different persons played the role of problem owner at different times. ECC started by playing this role and prescribed the use of a comprehensive QA-plan. Later Zwijsen became the new and definite problem owner. ECC is an educational software developing company, Zwijsen is an educational publisher, and they use completely different ways to control the quality of products. ECC provided, on behalf of POCO, the budget and expertise to design the framework of *Woordwijs.* ECC also realized all the software and filled, iteratively, the first module for grade level 4 in co-operation with Zwijsen. Filling in the contents for the other grade levels was Zwijsen's responsibility, after the project.

The project leader of ECC was responsible for the design and realization of the educational software. The project leader of Zwijsen was responsible for the design and realization of the contents: texts, pictures and audio segments.

Co-operation between ECC and Zwijsen was fine up to the acceptance procedures planned by ECC to be performed partially by Zwijsen. The acceptance procedure for the software was performed according the POCO QA-plan using a test plan and a formal function analysis. The acceptance procedure proved to be difficult for several reasons. First, it was difficult to accept the software, the responsibility of ECC, independently from the contents, the responsibility of Zwijsen the publisher. Second, the publisher had no experience with formal acceptance procedures of software based on approved design documents and a formal acceptance plan. This way of working is common in engineering but not for educational publishers in the Netherlands. Zwijsen was not used to applying formal acceptance procedures to its traditional educational products. It took much labor by ECC to convince Zwijsen management that acceptance of the "empty" software drivers and editors was necessary before all the contents could be filled in. A representative part of the contents should be sufficient. This was partly caused by the fact that the publisher was uncertain, at the time, about the market perspectives for *Woordwijs.*

This project was a small one (about 2 man year[1]). Therefore some explicit control steps were omitted with approval of the PM and the PL, see table 7.3.

1. The development ratio for the exercises for grade level 4, including all supporting software and drivers to fill in the contents is about (2x1600 man hours)/ 9 hours educational software = 355. Filling in of the exercises for the grade levels 5-8 was accomplished with a smaller ratio by the educational publisher Zwijsen. The total number of hours of all exercises for all grade levels is about 42 hours. A ratio of 100 - 150 can be estimated.

Phase	Way of controlling	Carried out/produced
product definition	consultancy report	yes
feasibility study	review of educational market perspectives	yes
	review of technical feasibility	yes
specification	review OSM/ED	yes
	internal review FD (including user interface design)	no, too simple
	external review FD	no, too simple
offer	review offer by SC	no, too simple
realisation	review technical design	no, too simple
	review system test plan	yes
	review system test report	yes
acceptance	review software and educational content	yes
field test	review field test by SC	yes
revision	review revision reports	no, too simple

Table 7.3: Produced reports for quality control according to the QA-plan

7.7 Conclusions

The *Woordwijs* package with the contents filled in for grade level 4 (module A) was completed successfully by ECC and Zwijsen. The modules B-E for grade levels 5-8 were published after this study took place. Zwijsen took over responsibility for the further marketing and completing of the contents. This was according to the aim of the POCO program. The package is sold in the Netherlands and Belgium as an additional component to the general reading method *Wie dit leest* and as exercises for teaching other reading methods.

Success, i.e. large scale use, of the use of *Woordwijs* cannot yet be reported. *Woordwijs* is a product of high quality. When it is used it will probably be effective. The product was judged positively in a consumer test[1] (SCEN, 1992), however, success is not guaranteed, although no or few competitors exist. The scale of implementation of *Woordwijs* in schools will depend on the advantages and profits the teachers expect for their work with students in the classroom. Students have to use the five modules over five full years. Many copies, each probably used by many children, have already been sold to schools (in the school year 1994/1995). In a general survey on use of educational software in primary education in the Netherlands (Brummelhuis and Plomp, 1993) four important bottle-necks are listed: too few computers, too little suitable software, too little expertise of teachers and management at a school, and the lack of time of teachers. These and other factors may be critical for the success of educational software, besides educational quality which can be measured experimentally. These factors can be controlled better for educational software projects for use in companies, see chapter 5 and 6. The whole POCO program (about 55 titles of educational software) has been evaluated thoroughly (POCO, 1992). It is claimed that by emphasizing the team approach and using a comprehensive QA-plan "the gap between the disciplines involved has been bridged" and that

1. Evaluated were the contents, the use in the classroom, the use by students, and technical characteristics.

"people from different disciplines learned to communicate".

To our opinion the educational quality of the product is good because:
- it is designed using relevant literature on learning Dutch as a second language;
- it is based on an existing reading method;
- a formative evaluation in classroom was carried out and revisions were made;
- a rich diversity of exercises was developed;
- the user interface including speech feedback is motivating for children;
- the teacher has extensive facilities to control and manage the use by many children.

Nevertheless, the success of this product depends not on the educational quality alone but on the product-market combination of it.

We will now answer the research questions for this project referring to the analysis described above. The implications of the answers are discussed briefly. See table 4.2 for the relationships between the questions and the aspects of the framework of Seligmann, Wijers and Sol (1989).

Question 1: How is the emphasis on instructional versus information technology aspects divided?
It is clear from the initiative for this project, see 7.2, that the emphasis was put primarily on instruction, but skills in information technology and software engineering were present at ECC at the end of the whole POCO program. The *Woordwijs* project was one of the last projects of the second development cycle of POCO. Within the POCO program, the QA-plan and the team of multidisciplinary, experienced staff, achieved a balanced approach on instructional versus information technology aspects. The emphasis was also put on instruction by the publisher Zwijsen. This product was developed for the open market. Therefore, professional marketing expertise was necessary to achieve success for this product. This is an important extra requirement for this project. Both ECC and Zwijsen were not capable of developing and marketing such a product as *Woordwijs* without the expertise and support of the other partner.

The way of thinking of the two partners ECC and Zwijsen introduced some conflicting interests. ECC's way of thinking was focussed on instruction, information technology, software engineering and quality assurance techniques, e.g. using the IEEE 730 standard. This would be satisfactory for projects such as those described in chapters 5 and 6 because commitment for use can be achieved by good organisation. Zwijsen is a well known publisher of traditional, educational products for schools. They are used to applying a different way of thinking, i.e. not based on engineering but more on media production. Although the prototype developed by ECC and filled with contents by both partners together were freely available to Zwijsen, Zwijsen hesitated about the market perspectives of *Woordwijs*. This hampered a common way of thinking on acceptation procedures between Zwijsen and ECC.

Question 2: Is the product under development considered as a tool and/or as a medium?
It was stated explicitly in the project documents that the motivation and the engagement of children of ethnic minorities should be enhanced. The diversity of the many types of exercises and the use of graphic animations for positive feedback after exercises illustrate the attitude of the designers of *Woordwijs* that the educational software package can also be considered to be a medium. The integration of media, e.g. using the same icons and illustrations on paper and on screen, also emphasizes the media approach. So does the use of speech output. The attractive illustrations in the books and software are consistent with this view. The method and the QA-plan used, both showing a professional engineering approach, focussed more on the tool approach than the medium approach. Thanks to the personal interest and skills of the

developing team a media oriented product was achieved.

Educational software is considered to be a medium for children while exercising and a tool for the teacher who has system features to compose exercises for the children and schedules for groups of children. For this project both the media and the tool approach had an important position in the way of thinking.

Question 3: Are the macro, meso, and micro levels of modelling distinguished explicitly?
The Object System Model, see 7.4, covers the macro and meso levels. The Educational Design covers the meso and micro levels. The design of the syntactic and lexical models of the user interface was given considerable attention. At the micro level the semantic, syntactic and lexical models were specified explicitly. A complicating factor for this modelling was the existence of the two partners involved. During the beginning of the project ECC devised first prototypes of the exercises, the structures and the contents. As the reading method *Wie dit leest* of publisher Zwijsen was chosen later and contracted as the educational basis the structures and of course the contents had to be revised by ECC; hence also several details of the user interface were revised. The secure modelling by ECC before Zwijsen joined the project provided good conditions for later changes and adjustments proposed by Zwijsen. The models were implemented in running prototypes of editors for filling in the contents. These editors were the final "models" of the different types of exercises, in which all early conceptual, semantic, syntactical, and lexical models were integrated and implemented. These editors were used by Zwijsen to fill in the hundreds of exercises to be used for the later grades. This implies that for this kind of subject matter and training by drill and practise exercises the design of the structures that have to be filled with contents consisting of words, sentences, pictures. and audio, should start with secure modelling.

Question 4: To what degree are different responsibilities within the development team defined and implemented?
Most of the tasks and responsibilities were defined explicitly in the QA-plan. The "mini" QA-plan for this project was "extracted" from the general POCO QA-plan. Within this project some interesting integrations of expertise were possible which resulted in a strong influence of a few persons. The project leader was an educational technologist with expertise in Dutch as a second language and she had several years experience in developing educational software for special education. She also has former experience of collaboration with the team of programmers who realized the educational software for *Woordwijs*.

When the partnership between ECC and Zwijsen was initiated the management styles of the two autonomous organizations had to be joined and merged; for most educational publishers developing (educational) software is an unknown discipline, as yet. Tasks and responsibilities are mostly defined and controlled *ad hoc*. At Zwijsen the project leader was experienced in her organisation and she had former experience in teaching, psycholinguistics of the Dutch language and scientific research. She founded at Zwijsen a small, temporary, educational software group to specify the contents and to use the editors provided by ECC to fill in the specified contents of the exercises. This group was responsible for the production of the final product as an integrated set of software, exercises and adjunct paper materials. The responsibilities were defined and implemented accurately because the partnership was based on contracts.

During the acceptance procedure by the developer and the publisher some questions arose about the different responsibilities for the structures of the exercises encaptured in the software (responsibility mainly of ECC) and for the contents of the exercises (responsibility of Zwijsen).

Indeed, there is a strong correlation between the characteristics of structure and content. Both influence the quality of the final product consisting of software (structure) and contents. In this project this problem was solved by negotiations and deliberation which took up much time. The implication of this result is that defining responsibilities is especially important when different disciplines are co-operating, as at ECC and Zwijsen.

Question 5: To what degree is communication realized explicitly between different disciplines?
The communication between disciplines was supervised completely by the ECC project leader. The critical communication between subject matter experts and educational technologist was no problem because the project leader incorporated these disciplines herself. Communication between the educational software developing team of ECC and the team of the publisher resulted in some problems initially. The project leader of the publisher, the authors of the reading method *Wie dit leest*, the persons who filled these contents using the computer editors, were all trained by the ECC development team. Due to lack of time and commercial hesitation by the publisher, it took some extra time before communication went well. Commercial hesitation by the publisher was the main reason for the initial delay in communication. Conflicting interests in the line function of the publisher and the project function of the other members were also reported.

The efficiency of the development process was rather high because communications were explicitly defined in the QA-plan and because some communication was not needed at all due to the broad experience of the project leader. The difficult communication between subject matter expert and educational designer was minimal in this project team. This was auspicious. The implication of the answer to this question is that communication can be improved by appointing people who are skilled in several disciplines needed for development.

Question 6: To what degree are the design and realization of the product separated into teams with different skills?
As required by the QA-plan design and realization and filling in the contents were separated. The design was done by the ECC design team. The technical design was done without formal approval documents. The detailed designs of the exercises were implemented directly on as many editors as exercise drivers. The drivers were used by the publisher to fill in the contents of the exercises for the first module (1 year) in co-operation with the developing team. For the next grade levels the filling of the exercises was carried out by the publisher's team without assistance by the ECC team. The development team prepared guidelines explicitly and procedures for the authors who had to fill in the contents of the exercises and to design and realize the icons and audiovisuals belonging to these contents.

Design and realization were separated. Realization was further separated into software and content realization, with separated responsibilities. This separation worked out positively thanks to the co-ordination of the two project leaders at ECC and Zwijsen. This project, as with many POCO projects, showed a special way of working which will not be found in projects were one partner is the "problem owner". The Dutch ministry was a problem owner at a distance by asking for improvements in learning of Dutch by minorities, the POCO program management owned the problem to deliver about 1000 hours of educational software within budget and time constraints; the publisher owned the problem to put a commercially successful product on the educational market. This characteristic of the project forced separation of the development process into more phases and components than necessary. Such a handicap can be compensated for by the motivation and skills of the project leaders.

Question 7: Is the project directed by one person who has a clear image in mind of the characteristics of the final product?
The ECC project leader was able to behave as a director. She was a both subject matter expert and an educational technologist, and she was experienced in designing educational software. She was also experienced in managing educational software development projects. She personally carried out most of the evaluation and quality assurance procedures during the design process. It was important that she was an expert in second language learning. Thus, she was able to overview different disciplines easily, and this is important to fullfil the role of director successfully. The project leader of the team of the publisher had to share responsibility for the contents of the exercises with the authors of the method *Wie dit leest*. This complicated the way of controlling of the filling in of hundreds of exercises. The project leader was able to play the role of director for each of the two stages of the project, the ECC and the Zwijsen stage. For the project as a whole this situation was not optimal, but given. This resulted in some ambiguity at the end of the project. This was caused by the built in separation between development and production, marketing and delivery for all POCO products. The POCO program was, per definition, a multi-partner program, introducing many of the co-ordination problems between autonomous partners. The implication of this answer is that the presence of one director for the whole project, including marketing is the ideal.

Question 8: To what degree are explicit quality assurance measures provided?
The ECC stage of this project was embedded in a strong professional management. The general POCO management, the ECC project leader and other staff involved in this project were all highly experienced in management. Within the context of the large POCO program of tens of educational software development projects a good infrastructure was available to manage the relatively small, 2 man years, project *Woordwijs*. Without such an infrastructure of a "courseware factory" it would be difficult to develop such a product with so few employees and because of the large experience within the organisation, the management could be flexible enough to decide to deviate from the requirements and the guidelines of the QA-plan. The management style at Zwijsen was different and characteristic of professional publishers. The adaptation of the way of working and controlling of both partners was mentioned as one of the difficult aspects of this project. This problem was resolved by negotiating in a commercial context.

As this product was developed for an open market a formative evaluation using eight students was carried out. The results were rather positive. Only a few improvements were advised, mainly concerning the user interface and student control. Of course it is not guaranteed that the product will be a commercial success. It will take some years to sell modules for the subsequent grade levels.

Specification of the subject matter was not the key problem of this project. The risks of the product seem to be concentrated in the quality of the user interface as perceived by the students and in the balance of the attractive presentation of exercises and the variation of styles of exercises and the media used.

The project leader of the development team had a strong vision of the aims and the necessary instructional and motivational quality. She had a highly relevant blend of expertise and skills to demonstrate leadership at the levels of management, educational design and subject matter. Most of the required means for quality control were carried out.

The benefits of the development method and the QA-plan especially tuned for educational software development were clear. The method shares phases and concepts with the new theory of chapter 4, including the macro, meso, micro levels.

An implication of this answer is that the use of a quality assurance method as presented in the new theory of chapter 4, is very important for the way of controlling.

8. CONCLUSIONS

In chapter 1 a general discussion concerning educational software development led to questions about how communication between disciplines should be carried out; how should conceptual and functional design be carried out; how should the leading role of the project manager be implemented and which management style should be chosen. A general research question was formulated to propose and validate criteria for the better organization of multimedia educational software development. A new theory for development was presented in chapter 4. Eight questions were formulated as criteria for evaluation. The research questions were answered for three case studies and the implications of these answers were discussed in chapters 5 to 7.

Three projects, where the use of an educational product was achieved for 1000 students or more were chosen for the case studies, multimedia is applied in different forms in the products of the projects. The first case study describes an in-house development project at a large bank. The second describes an out-house project designed for the aircraft industry. The third describes a multi-partner consumer oriented project for primary schools, initiated by the government and finished, with a commercial product, by an educational publisher.

The answers to the research questions of the case studies are surveyed and discussed in section 8.1. In section 8.2 the new theory is evaluated using these answers. In section 8.3, general conclusions are drawn about applying the new theory for developing educational software. In section 8.4, future research is proposed.

8.1 Answers to the research questions

The case studies in chapters 5-7 are structured according to the ways of thinking, modelling, working and controlling. The conclusions are described in terms of the answers to the research questions formed in chapter 4 and some implications. The conclusions about the three case studies are evaluated in this chapter and compared to discuss the new theory derived from the literature survey and the analysis of the inductive cases given in chapter 3.

The cases described have rather different general characteristics. The subject matter is very different, consisting of commercial training of bank employees (*BZD*), technical training of aircraft maintenance technicians (*MT*) and second language exercises for children of ethnic minorities (*Woordwijs*). The didactical style for all cases is basically the same, drill and practice. In the *BZD* and Fokker *MT* cases aimed at training professionals, the didactical style of drill and practice is merged with the style of simulation. In the *Woordwijs* case no simulation of real situations is provided to exercise words and sentences. In all of the cases the educational software substitutes for other modes of instruction, so it can be classified as primary educational software (cf. 1.2).

The *BZD* case was an in-house project in which only a few parts of the design and realization were carried out by external partners under contract. The *MT* case was an out-house project in which many parts of the design and almost all of the realization were carried out by an external partner under the control of the problem owner. The *Woordwijs* project had a special organizational structure called "multi-partner". The final problem owner was the educational publisher marketing the product. The initiative to develop the product was first taken by an "external" partner controlled by the Ministry of Science and Education.

CONCLUSIONS

All three products were delivered successfully and are being used by many students in the target groups.

The answers to the research questions are summarized in table 8.1.

Aspect of framework	Research questions	*BZD* in-house/1.6 my	*MT* out-house/30 my	*Woordwijs* multi-partner/2my
way of thinking	1. how is the emphasis on *instructional* versus information technology (I.T.) aspects divided?	instructional: yes (2) I.T.: no (0)	instructional: yes (2) I.T.: yes (2)	instructional: yes (2) I.T.: yes (2)
	2. is the product under development considered as a tool and/or as a medium?	tool: yes (2) medium: no (0)	tool: yes (2) medium: yes (2)	tool: yes for teacher (2) medium: yes for student (1)
way of modelling	3. are the macro, meso, and micro levels of modelling distinguished explicitly?	macro: yes (2) meso: partly (1) micro: no (0)	macro: yes (2) meso: yes (2) micro: yes (2)	macro: yes (2) meso: yes (2) micro: yes (1)
way of working	4. to what degree are different responsibilities within the development team defined and implemented?	defined: yes (1) implemented: partly (1)	defined: yes (2) implemented: yes (2)	defined: yes (2) implemented: yes (1)
	5. To what degree is communication realized explicitly between different disciplines?	less (0)	yes (2)	yes but difficult between partners (1)
	6.to what degree are the design and realization of the product separated into teams with different skills?	separation: no (0)	separation; yes (2)	separation: yes (1)
way of controlling	7. is the project directed by one person who has a clear image in mind of the characteristics of the final product?	no (0)	yes (2)	two partners involved: two persons (1)
	8. to what degree are explicit quality assurance measures provided?	partly (1)	extensively (2)	extensively (2)
way of supporting	no questions	-	-	-

Table 8.1: Summary of research questions and answers.The subjective scores for the answers are: 0 means no or barely relevant; 1 means moderately positive; 2 means very positive.

Question 1. How is the emphasis on instructional versus information technology aspects divided?

In the three cases a strong emphasis was given to instruction, however, it seems to be difficult to create a balance of emphasis for both instruction and information technology. In the *BZD* project this balance was difficult to realize because limited professional expertise on information technology was available internally and was not obtained externally. As a consequence the interactivity of the product was too low for this subject matter and this target group, see 5.7. In the *MT* project the balance was obtained by calling in a third party, experienced in professional educational software design and realization. In the *Woordwijs* project all the expertise, including subject matter expertise, needed for the balance was available within the one team, much of it embodied in one person. The *BZD* product did not get a positive educational evaluation, see section 5.7, and maintainability is not good due to the lack of technical designs. For these reasons, in our opinion, use can be expected to decrease in the future, although it is used on a large scale now. For both the other projects the educational evaluation was positive. Use of the *MT* product is guaranteed and the product is easily maintained as has already been proved. Large scale use[1] of the'*Woordwijs* product was not certain at the time this study took place because the product-market combination could not be guaranteed. These answers imply that a strong emphasis should be given to both the instructional and the information technology aspects of educational software.

In the *BZD* and *MT* cases much emphasis was put on subject matter analysis of the functions/tasks the target group need to be trained in. This task analysis was most difficult for the *BZD* case. A special meso modelling technique, task analysis matrix, was applied to acquire approved descriptions of the functions and tasks experienced bank employees have to perform. In the *MT* case the procedures to be trained, the subject matter, were already described, but not didactically, in maintenance manuals produced by the aircraft engineers and in training specifications of existing maintenance courses. In the *Woordwijs* case accurate specifications of the subject matter were available in reports and other learning materials. In the *BZD* case three development processes had to be carried out simultaneously: specifying the subject matter (1), designing a didactic solution (2) and using new media (3). In the *MT* case the subject matter was known from existing courses but the didactics (2) and the media used (3) had to be designed from scratch. In the *Woordwijs* case the subject matter and possible didactic solutions were well described in reports and the literature but the media used (3) had to be designed from scratch. This observation gives, in our opinion, a reason why in the *BZD* project the team did not place an emphasize on information technology aspects. If the subject matter is not yet known instructional aspects become, relatively, so important that a real danger exists of underestimating the information technology aspects. This implies that emphasis on both aspects, instruction and information technology, is important but that what the balance needed depends on the characteristics of the specific project.

In the large *MT* case, about 30 man years, a balance was achieved by the teamwork of many team members from different disciplines and organizations; the project leader demanded, and controlled, an emphasis on different disciplines. In the "medium-sized" *Woordwijs* case, about 2 man years, the balance was achieved by a small team from two partners in which some employees had experience in several disciplines; the project leader was personally experienced in different disciplines. In the small *BZD* case, about 1.6 man years, expertise in information technology was less available in the team and this was not compensated for by using an external partner; the project leader was not experienced in disciplines other than instruction technology and general management. These answers imply that an emphasis on the aspects instruction and information technology can be given in several ways, but the way of thinking should support this and the required disciplines should be available. This finding is supported by the vision of

1. If *Woordwijs* is used over 5 years in 30 schools by 30 children each this would result in 900 users. This can not yet be classified as a success for this kind of product in our opinion.

Bork et al. (1992) that for all necessary disciplines, and he mentions a long list, excellent professional expertise should be available. Educational common sense and zeal are insufficient.

Question 2: Is the product under development considered as a tool and/or as a medium?
In the *BZD* case the interaction is designed in a command style based on cognitive reasoning about a simulation. Many learning situations are composed by simulating communication with clients at the bank counter. The interaction of the student with the simulated client is completely text based using menus. The introductory video tapes illustrate and prepare the simulations off-line. The simulations do not have a realistic "drama" component in the way real communication with clients has. The simulations are based on a standard model, script, for diagnostic dialogues and meta discussions about how to communicate with the client are embedded in the design of this model The interaction is static and presents small predefined sentences to the student. The student is forced to behave as a passive spectator. Pleasure and the direct engagement of the student is not triggered by the interaction style used. In this case educational software is not considered to be a medium for students but a tool. Within the simulations the concept of a "medium" for the student is intended but this is realized and shaped as a tool. This answer implies that the characteristics of the educational solution produced depend directly on the general way of thinking on education. The designers of the *BZD* product had a solution in mind using exercises and cognitive simulations of dialogues with clients. Such simulations were developed nicely; however, the question is do these simulations offer sufficient engagement for effective training? According to our opinion and the outcome of the evaluation by Driessen, section 5.7, the answer is not positive.

In the *MT* case different media are integrated in the interaction of the simulations. The procedures for maintenance are presented and trained by menus, task lists and direct manipulation of displayed devices, figure 6.4, of aircraft. Interaction is personalized for the student by audio explanations and audio feedback to improve the direct engagement of the student; this is essential in such areas as aircraft maintenance where high standards of work and concentration on the job in hand are required. In this case educational software can be considered both as a medium and as a tool. The evaluation of the project and the product, and the reported satisfaction of instructors and customers are positive. When engagement is aimed at it is possible to achieve it by direct interaction and audiovisuals; however, the price is high as can be seen from the development ratios for the *MT* and *BZD* cases.

In the *Woordwijs* case the subject matter and several didactic solutions were relatively well known within the team; much emphasis was given to *how* the interaction should be carried out. The goal was to motivate the students from ethnic minorities by good media use. A graphic user interface is applied consequently. Special graphics are used for some types of exercises to create an attractive user interface for the young student. Although the functionality in the interaction is realized consequently, a rich variation of exercise types is introduced providing a stimulating and motivating environment for the student. After finishing a series of exercises an attractive animation is presented to the student to increase engagement in the learning process. This product can be seen as a medium rather than as a tool for the student. This emphasis on the medium aspects of educational software can be understood because second language learning by children has to be made very attractive to fascinate and motivate them to continue learning over a long period. Pleasure and amusement are explicit aims for this kind of educational software. The results of this case study show that it is possible to develop a motivating and engaging product with time and budget limits. From formative evaluation (SCEN, 1992) and according to experts, *Woordwijs* is a nice and usable product. Future success in the educational market is not, however, guaranteed. It was easier to forecast the scale of use of the tailor-made *BZD* and *MT* products. One can say that the use of these products was in a way compulsory for special target groups. Their development costs do not have to be reclaimed directly as with *Woordwijs*. This implies that considering a product under development as a medium can contribute largely to the quality, but does not as yet guarantee successful use.

Question 3: Are the macro, meso, and micro levels of modelling distinguished explicitly?
In the *BZD* project careful models made only at the macro and meso levels. Modelling at the micro level was not carried out using a method as mentioned in chapter 4. This was caused by less emphasis being put on interaction design. In the *MT* and *Woordwijs* projects the macro, meso and micro level educational models were made explicitly. Semantic, syntactic and lexical models were also made to specify the user interface. In the *Woordwijs* project two kinds of educational products were developed: the drivers and editors for the exercises in which the meso and micro models are implemented implicitly, and the contents of hundreds of exercises.Although in the *MT* and *Woordwijs* projects the macro, meso, and micro levels of modelling were distinguished explicitly it appeared difficult to carry out the design and realization of the user interface. The answers for the three cases imply that the distinction of the three levels of modelling is necessary but not sufficient. When the distinction of the micro level is not made, as in the *BZD* case, it seems difficult to design attractive interaction. When models at the micro level are made, as in the *MT* and *Woordwijs* cases, it seems as yet, difficult to design attractive interaction when the required expertise in media design is not available.

Question 4: To what degree are different responsibilities within the development team defined and implemented?
In the *BZD* case, tasks and responsibilities were defined clearly according to the general project management method. As this method was not suited for software development tasks responsibilities for the technical disciplines were not defined or implemented. During the project definition phase responsibilities were defined and implemented clearly. From the design phase on, as persons from other departments had to be involved, prescribed responsibilities and commitment weakened and the management style became informal and flexible. This answer implies that defining responsibilities should be followed by getting commitment to implement them. If responsibilities become less formal the project leader should be able to evaluate personally, and continuously the quality of the work of the members of the team. The project leader did not have subject matter expertise. It also implies that it is almost impossible to define and implement responsibilities when the project leader does not have an overview of the disciplines involved.

In the *MT* case tasks and responsibilities were defined and implemented clearly. The contract between the partners and the use of a well known QA-plan for educational software development supported this. This implies that it is important that the organisation is experienced in defining and implementing responsibilities. The team had this experience.

In the *Woordwijs* case all responsibilities were defined very clearly because it was one of a series of projects for educational software development within the POCO program. A generic QA-plan, used by all POCO projects, was applied. The responsibilities of the publisher were also described carefully. It was, however, not easy to implement these because the publisher did not have a tradition of co-operating in the area of educational software development. This implies that the kind of relationship between partners influences the way responsibilities can be implemented. In this project no partner was the exclusive problem owner looking after implementation of *all* responsibilities.

Question 5: To what degree is communication realized explicitly between different disciplines?
In the *BZD* case communication and collaboration between subject matter experts and educational technologists was impeded by the lack of real common interest between different independent departments within one company. Therefore, communication and collaboration were not optimal despite high levels of personal skills. During both the design phase and the realization phase no professional software engineers were involved; educational software designers and authoring language programmers educated in different non-technical disciplines were used. The project management method used was too general, and did not give checklists with tasks, responsibilities, and required skills, and did not prescribe explicitly special documents for internal communication on the design. This implies that communication can be aimed at, but implementing it is difficult when full commitment by all partners, or some necessary expertise is missing.

In the *MT* case the development method used provided exhaustive guidelines for, among others, collaboration and communication between disciplines. The problem owner demanded that all team members of BSO/Origin worked together at the Maintenance Training Centre. The project leader of the BSO/Origin team was assigned explicitly to evaluate and improve the QA-plan continuously during the project. Communication, reviews and problem reports were organized rather formally according to the current QA-plan. The communication between disciplines was analyzed using problem reports. Several times members of the team were trained explicitly to support colleagues with other disciplines to solve temporary capacity problems for different disciplines. An analysis of the problem reports (Schouw, 1992) shows that most of the communication problems were detected between subject matter experts and educational technologists. In our opinion this was caused by the fact that the subject matter experts, full-time instructors, were not real team members. This implies that the communication between subject matter experts and educational designers is cumbersome in projects where specific subject matter experts are indispensable both to their teaching job and the development team.

In the *Woordwijs* case a comprehensive development method was also used to define how communication between disciplines should be implemented. Although it was a relatively small project of about 3 man years these guidelines were used explicitly when considered necessary by the POCO quality controller and as the responsibilities of employees were specified explicitly, the advantage of having several disciplines and skills embodied in one person could be utilized effectively. This answer implies that team members having significant experience in several disciplines can improve communication within the team.

Question 6: To what degree are the design and realization of the product separated into teams with different skills?
In the *MT* and *Woordwijs* projects the complete meso and micro models were documented, reviewed and approved formally. The realization teams used these written designs, some prototypes of interaction details and comments from the design team as input. Separation was according to our theory. In the *MT* case separation decreased at the end of the project, caused by problems with the availability of man power. This caused a measured decrease of productivity. In the *BZD* project the educational design process yielded an extensive description of the subject matter as a meso model. This design was used as the draft for the textbook belonging to the educational software. The draft of the textbook was the input for the realization team of the educational software. The realization team first had to design the technical models at the micro level before realization could be started. This is not the kind of separation the theory describes. The realization team worked somewhat independently of the

design team, however subject matter experts had to accept and approve the models at the micro level so that the content was consistent with the models at the meso level. This approval was an informal process in many steps. Given this, design and realization were not separated in time nor carried out by teams with different skills. Another separation took place: designing the macro and meso models from designing the micro models and realizing of the total design. This answer implies that when separation between design and realization is not prescribed, cf. *BZD* case, by the way of working nor planned accurately, it will take place in an unpredictable way without formal approval of a complete design. This is not acceptable when quality assurance is aimed at.

Question 7: Is the project directed by one person who has a clear image in mind of the characteristics of the final product?
The *BZD* case was directed by an educational technologist with no expertise on the subject matter of the product. Therefore, the project leader could only partially and indirectly, not by personal expertise, supervise the quality control of the design and the final product and because of conflicting interests from line and project functions, he had some problems in acquiring full commitment from the subject matter experts working for him. A project leader with a multi-disciplinary profile, including expertise on banking, would probably have chosen something other than "cognitive" simulations of dialogues with clients. This project leader can not be considered to be the intellectual creator and owner of the educational software product. This answer implies that a project leader having, independently or in close consultation with the formal problem owner, no clear image in mind of the final product, will solve his own problem as project leader. His problem differs mostly from the "real" educational problem. The theory prescribes that the project leader plays the role of the creative and managing director as in film making.

In the *MT* case the role of director was played by the project leader of the Fokker CBT team. He had a clear image in mind of the characteristics of the final product and was used to working according to the high quality standards required in the world of aircraft manufacturing. Although the subject matter experts did not have experience with designing educational software the project leader succeeded in getting them involved with high commitment. The responsibilities of the Fokker CBT team and the BSO/Origin team were very clear. The Fokker project leader was assisted by the BSO/Origin project leader in achieving a high quality educational software product. This answer implies that delegating and sharing a part of the responsibility of the project leader by contract is possible if this is done accurately and if the project leader continues to control the progress of the project and the quality of the product.

In the *Woordwijs* case the project leader was able to supervise many aspects of the design and realization because she was experienced in several of the skills needed in the team. She was able to evaluate and control all the relevant aspects yielding the overall quality of the educational software, permanently and personally. She played the role of director clearly. When the collaboration with the educational publisher started she was able to share responsibility with the project leader on behalf of the publisher and to transfer her responsibility later when the publisher continued filling the contents of the next hundreds of exercises independently from ECC. This answer implies that delegating and sharing the responsibility is also feasible under more complex conditions if this is done carefully and as formally as possible, and if the project leader continues to control the progress of the project and the quality of the product.

Question 8: To what degree are explicit quality assurance measures provided?
In the *BZD* case a general project management method was used. This method is domain independent and does not provide guidelines or phasing schedules for software development and hence not for educational software development. This general method failed when expertise concerning educational technology and software development was required to play a dominant role during the project; when this happened project management became informal. The project was, however, completed using perseverance and strong general management, but with little emphasis being given to important aspects of software engineering and user interface design. It was difficult to specify and maintain a *detailed* QA-plan for the realization phases of the project. This answer implies that it is difficult to control a project without explicit quality assurance techniques for all disciplines involved.

In the *MT* case it was expressed explicitly by the problem owner that project management of developing multimedia educational software is very difficult and that such expertise was not available internally. One of the main reasons to call in the third party for the project was their experience in project management of educational software projects including QA methods. During all phases of the project, project management and QA were carried out extensively and carefully. This answer implies that quality assurance can give positive results at least if the problem owner and the project leader are aware of how critical it can be for success, and if funds are available for it.

In the *Woordwijs* case, a project of the very large POCO program of educational software projects, a specialized method for management of educational software development including quality assurance was used. This was a project of the second cycle of the POCO program and experience from many former projects was available for the QA-plan and among the employees of the management team. In this project it became difficult to sustain management during the acceptance procedure when the development team had to co-operate with the team of the publisher. Both teams had a different management style. This implies that being experienced with using quality assurance techniques works out positively, but that they should be accepted by internal members and external partners.

8.2 Discussion of the new theory

The new theory is summarized in 4.7 as follows.
• An explicit, practical problem with education or skills should exist and be identified. An organisation or person should exist who is able to behave as the owner of a problem (T1). The problem owner and the leader of the educational software development project should focus explicitly on the aspects (T2) education, information technology and the media used (way of thinking).
• To model the problem and possible solutions macro, meso, and micro levels should be distinguished explicitly (T3). Specifying the user interface (T4) should be carried out professionally (way of modelling).
• Commitment of all persons involved should be based on clear definition and assignment of responsibilities. The required skills and disciplines should be represented within the team and the necessary communication within the team should be anticipated. Designing and realization (T5) should be carried out by separated teams (way of working).
• The project leader should be able to behave as a director (T6) having a clear image of the final product and having senior experience to overview all disciplines involved in the development team. Professional quality assurance measures (T7) are necessary (way of controlling).
• Tools should support the transference of thought (T8), "passing the baton", within the team (way of supporting).

This theory prescribes a general, multi-disciplinary, approach for developing educational software by teams. This theory proposes indications and guidelines for the start-up phase of projects. This is sufficient because the manyfold disappointments of educational software projects and products are not caused by problems with the details of a design process but by misconceptions of the function and the use of educational software in context[1]. Focussing on the best *educational* quality can distract from other aspects more important for successful *use*.

From the answers to the research questions we can conclude the following concerning the elements T1-T8 of the theory.

Way of thinking

In one project (*MT*) the problem and its owner were clearly evident. In the two other projects the owners of the problem were not known clearly for different reasons. In the *BZD* project it is not clear who was the real problem owner: the anonymous community of local banks requiring better functioning of their employees; the Corporate Sector director, the responsible principal for the project; or the project leader himself who was motivated to show the expertise of his department? It was almost impossible to implement commitment by all participants. This problem could have been prevented if the "problem owner" was a person who was formally responsible for the level of expertise of all counter employees and if the project leader had another hierarchical position within the organisation. In the multi-partner *Woordwijs* project several owners of different problems existed: the Ministry providing the funds and willing to stimulate the use of information technology; ECC willing to manage the POCO program within time and budget limits; and Zwijsen willing to sell an educational software package fitting an existing reading method. This problem could not be prevented by the built-in multi-partner characteristics of the POCO program as a whole. The conclusion at this point is that it is very important to recognize educational and skill problems and their owners accurately and explicitly at the start of a project. *(T1)* of the theory is *not rejected.*

The second point of this aspect of the theory is the emphasis on educational, information technology, and media aspects. From the case studies it was evident that all projects put much emphasis on the educational aspect with one significant difference. In the *BZD* and the *MT* projects the subject matter, procedural work by humans, was not explicitly known before the project started and had to be acquired by profound analysis. This demanded high emphasis on educational aspects at the cost of the other aspects. When the emphasis on information technology and media aspects was lower, caused by either lack of interest or lack of expertise, the product was less motivating and engaging according to student evaluations. In the *BZD* project media selection was assigned to an external consultant. His report mentioned that media selection was not carried out using detailed knowledge of subject matter. A traditional approach according Gagné and Briggs was used, see figure 5.6. Media selection aimed at "simple, low budget solutions for educational software which can be realized using simple authoring tools[2]". This is, in our opinion, not a professional media approach because of the too optimistic expectations about the costs of media use. The emphasis on educational, information technology and media aspects required depends on the characteristics of the product. The

1. The use and "success" of educational software packages depends on the choices made and designs for *all* levels macro, meso, and micro of the theory. If use is compulsory, as with most tailor-made projects, large scale use does not depend only on the educational quality, cf the *BZD* case study. If the product is made for the open market the unpredictable laws of consumer (institutes, teachers, and students) preference come into play. Mostly the best marketing-product combination 'wins', not the best product.
2. Quotation from the final internal report on media selection.

conclusion is that *(T2)* of the theory is *rejected*; the three aspects mentioned are important, cf the negative consequences of missing one aspect in the *BZD* case, but not complete. It should be mentioned that the way of thinking required also dependents on external aspects, such as whether a product will be used compulsorily, whether it is a tailor-made product, or not, whether it is a market oriented product; in both cases the return of investment is totally different which may influence the way of thinking. These aspects may be even more important than thoroughly the three mentioned in (T2) of the theory.

Way of modelling

Distinction of the macro, meso, and micro levels of modelling can be recognized in the *MT* and *Woordwijs* projects. Both projects were carried out using a different method derived from the HOEP method, see 2.2.7, in which software engineering techniques are prescribed. At the micro level the semantic, syntactic and lexical models can be recognized, be it implicitly. The techniques used for modelling the user interface are not professional but *ad hoc*. At the micro level, a professional way of modelling is necessary; it requires expertise on information technology and media design. In the *MT* case the internal evaluation report mentioned that techniques for specifying and realizing the user interface could be improved. From the case studies as well as from the survey in chapter 2 it is evident that it is not critical which notation techniques for modelling, e.g. graphic diagrams or tables, are used provided that they are suited to the level of modelling. The aim should always be their power to structure, express, and transfer the thoughts of the designers. The conclusion is that *(T3) is not rejected*, but that *(T4) is rejected*, because it depends on the kind of interaction designed. The BZD product is not highly interactive; therefore it could be designed and realized without professional user interface techniques.

Way of working

The way of working is not an independent aspect; the other aspects have to be tuned to and recognized in the way of working. It is evident from the answers to the research questions that commitment and communication between disciplines is difficult to achieve without explicit definition and control of responsibilities. From the large out-house project it is evident that it took much attention and energy from the project leader and his assistant to keep responsibilities and commitment at a satisfactory level, especially when the available man power for any discipline was not adequate. In this project the project leader and his assistant had an overview of all disciplines involved. During the large project (*MT*) design errors and revisions were recorded on forms. Continuous evaluation of these forms by the project management revealed where and when communication could be improved. The different disciplines involved in the large project and explicitly mentioned in the project documents were project management, subject matter expertise, educational design, software engineering and media expertise. This is in compliance with the list prescribed by the theory. Communication problems can also be caused by conflicting interests of employees who are not a formal member of the team. In two case studies the subject matter experts had a line function outside the project while they were called on for subject matter analysis and design. From the *Woordwijs* project it is evident that for smaller projects disciplines and responsibilities can be embodied in fewer persons, if explicit descriptions of required disciplines and responsibilities are provided. When different, independent, partners are involved responsibilities should be described in contracts. When required disciplines are not represented within the team several necessary phases and activities are relinquished or not carried out at all. This holds especially for the discipline of software engineering that often is considered to be superfluous when authoring tools are used. The separation of design and realization worked out positively according to the analysis of the *MT* and the *Woordwijs* projects. The conclusion is that definition and implementation of responsibilities is very important for team-based projects and that communication between disciplines has to be supported explicitly. This *part of (T5) is not rejected.* Separation of design and realization is necessary for large projects, but not for smaller projects for educational software with low interactivity[1]. Thus, this *part of (T5) is rejected.*

Way of controlling
From the answers to the research questions can be concluded that both for large, here about 30 man years, and for small, about 1-2 man years, projects the project leader should have a clear image in mind of the product. For one of the cases the project leader, an educational designer, had no expertise of the subject matter nor was he able to evaluate the designs personally as a student of the target group. As a consequence he could not control the details of the contributions from external partners, e.g. the external consultant who carried out the media selection. Therefore, this media selection was limited to a general level without regard for the details of the subject matter. This hampers the project leader in evaluating the feasibility of the proposed media selection. The project leaders of two projects were able to play the role of "director" to some extent. They had expertise and power to decide on schedules, budgets, appointment of team members, and assignment of internal and external extra tasks. From positive and negative examples *(T6) is not rejected.*

From the case studies it is evident that using and updating a formal quality assurance plan covering all disciplines is worthwhile, but expensive. The use of a quality assurance plan seems to be easier with out-house projects, as it is usual to describe goals, requirements, responsibilities and other conditions formally in contracts. In projects without main external contractors it may be difficult to make hard contracts between internal departments. This can influence negatively both internal commitment and implementing quality assurance. Anyway, quality assurance does not work when disciplines and skills are not available and can not be contracted from outside. Quality assurance has its price; its success depends on the way of thinking including an emphasis given to "quality" of the product and on available budgets. It can be concluded that *(T7) is not rejected.*

Way of supporting
The theory proposes that the way of supporting is not critical, and is to some extent invariant, for the success of a project. In the cases three different prototyping and programming tools were used. They were chosen for project dependent reasons. In one project the authoring language Tencore was chosen because this language was already used for other projects and because for this project the interactions would be text-based. In the second project the authoring system Authorware Professional was chosen because of its features to present multi-media and its compatibility for MS-Dos and Macintosh platforms. In the third project the programming system Visual Basic was chosen because experienced programmers were available and because it was, at the time, a good system for programming applications for MS Windows, one of the standards chosen for the whole POCO program. The systems were used for prototyping first and later for realizing the final product. The last two systems were used by programmers with expertise on software engineering. Tencore was used by educational programmers without expertise in software engineering. It can be concluded that only

1. Small *one person* projects carried out at universities using authoring languages, Sherwood and Andersen (1993), may develop highly interactive products without separation of design and realization.

prototyping and programming were supported by automated tools and that only the programmers of a project used the tools and showed the prototypes to the other team members. No general conclusion can be drawn about which tool should be chosen in a project. The theory states that the purpose of tools is to support the transference of thought within the team. Only prototyping was used by the teams of the case studies. Only programmers used the prototyping tools. This means that *(T8) is not rejected.*

The theory for developing educational software distinguishes five aspects each worked out into concepts and principles for development in teams. These concepts and principles attempt to give indications for better organization of development, see section 1.7. The theory does not claim more than that. A theory for development of products with an essential media component can hardly claim more than to present a paradigm for *preparing* a project. Depending on the domain, the concepts and principles of the theory can be used as guidelines for the preparation and start-up of a new development project. Depending on the subject area and the organisational context and constraints a quality assurance plan should be proposed and approved. The most important guidelines concern the management:

- the recognition of the function of the problem owner;
- the necessity to generate commitment;
- the need to appoint a project leader who is able to play the role of central director for all disciplines;
- the definition of responsibilities as a basis for all assignments.

One other main guideline concerns

- the need to provoke the engagement of the student by the media used.

8.3 Consequences for developing educational software

The results of this study give criteria for better organisation of multimedia educational software development. "Better organisation" includes, in the context of this study, improvement of *one or more* aspects of the framework used. The most important consequence of the results is that the way of thinking before and during a project is most significant for success and not the ways of modelling, working, controlling, and supporting. The *way of thinking* determines which disciplines, skills, and responsibilities are allocated and assigned to a project. In other words "good methods" do not work without a suitable way of thinking and, thus, the way of thinking should be part of a method. The guidelines mentioned at the end of section 8.2 provide for a practical way of thinking.

1. recognition of the function of the **problem owner**. This seems a trivial point, but it is neglected for several reasons in some of the case studies, and in the methods reviewed in section 2.2. The authority of the problem owner and the outline of his problem should be made clear on paper. The problem can be educational, but it can also be e.g. logistical, concerning the providing of training, or commercial, concerning the selling of training. The problem owner should know his role during the project. If the problem and its owner cannot be specified a NOGO decision for the project should be made.
2. the necessity to generate **commitment**. This seems to be a trivial point. It may be difficult to get commitment from different departments within one institute or organisation. Contracting third parties can solve this problem (Kvavik et al., 1994).
3. the need to appoint a **project leader** who is able to play the role of central director for all disciplines. He should be able to evaluate the quality of the design at all stages. This is a new guideline to some extent related to Bork's vision that "only the best experts are good enough for developing educational software (Bork, 1984).

4. the definition of **responsibilities** as a basis for all assignments. It is better to use responsibilities than (intermediate) products as directives for assignments of team members. Thus, selection and appointment can dependent on relevant skills more directly.
5. the need to provoke the **engagement** of the student by the media used. This can be achieved by assigning one person skilled in media design who is responsible for the concept and the details of the user interface engaging the target group. S/he can arrange interviews of people working on the intermediate designs and prototypes. "Well designed multimedia will draw in the learner, motivate them to continue their interaction and help them to accomplish their goals without distraction" (Jacques, Preece and Carey, 1995), however, when designing multimedia one should avoid 'overkill' (Van Aalst, 1995a,b; Hoogeveen, 1994).

The management of projects can be improved by applying the conclusions of section 8.2 during the preparation of new projects; for experienced, professional educational software development departments this means that chapter 4 and one of the test cases must be studied before the conclusions can be applied directly to their own context. The conclusions presented here cannot be used as an instant checklist. The most important advise for a professional is to start to compare the way of thinking described here with the way of thinking in their project, and to change their way of thinking if necessary.

In this study the concept of "educational quality" is intentionally not defined. Measuring the quality of educational software regards only a part of the problem (NCP, 1992b). Other "qualities", (e.g. international standardization of training, availability independent of time, and scale of use) of educational software are as important as educational quality. As an example we can compare the five case studies discussed. One study shows a product, see section 3.1, with good design at the meso and micro level. This product was never used because marketing and distribution failed for "political" reasons. Another study shows a product for TV, see section 3.2, that was never evaluated formatively but that was broadcast successfully many times and followed by tens of thousands of students who bought the course materials. A "semi-market" oriented product, see chapter 5, showed a simple educational and interactive design using text-based simulations, but it was a success because it is used by thousands of in-house students. A tailor-made product, see chapter 6, is nice and is now used, compulsorily, by thousands of students all over the world. A last, market-oriented product, see chapter 7, had a positive formative evaluation but large scale use in the market is as yet unsure.

8.4 Further research

The inductive research method of this study was based on a literature survey, the study of two inductive cases and the investigation of three test cases. The validity of the results of this study is influenced by the choice of the two inductive cases and the three test cases. The inductive cases forced the introduction of software engineering and professional project management principles (first inductive case) and TV and film making principles (second inductive case) to the theory. Two of the test cases use development methods that are descendents of the same method. More projects could be used for case studies. Another limitation of the research method is that all the case studies were carried out after completion of the project by studying project documents and interviewing the project leader and a few other persons. Information collected in this way represents the subjective opinions of people involved in the project. Further investigation is needed to detail the theory and to develop it into a new method that is used in real, multimedia, educational software development projects. Further investigation should include the topics:

• development and implementation processes of multimedia products by other disciplines (Netherlands Design Institute, 1994) in different application domains, e.g. industrial design engineering, informational TV (Heines, 1984), multimedia entertainment products and games, CD-I products, multimedia retrieval systems for marketing and sales (Hoogeveen, 1994). The way of thinking used, the comparison of design paradigms, the management techniques used, the communication between disciplines, and the role of the project leader may be important research topics.
• critical factors for actual use of multimedia educational software (NCP, 1990). Educational quality of a product, in whatever way it is defined and measured, is not the only requirement for actual use. How can marketing be involved in educational software design for a market where consumers decide on buying educational software.
• formulation and evaluation, using new projects, of a prescriptive method based on a blend of existing professional methods and the new theory presented here. Any enhanced method should emphasize means to improve quality assurance.
• the paradigm of "learner-centered" information systems development (Soloway et al., 1994). An information system is used by a professional who is supposed to be an expert. This professional is - should be - constantly learning how to use information (technology) better. Information systems development should move from technology-centered, in the past, and user-centered, today, into learner-centered, in the near future.

9. REFERENCES

Aarnoutse, C. and J. van de Wouw, *Wie dit leest Handleiding A*, Uitgeverij Zwijsen b.v., Tilburg, 1990.

Agresti, W. W., *New Paradigms for Software Development*, Elsevier Science Publishers B.V. (North Holland), 1-14, 1986.

Andersen, D.M., and B.A. Sherwood, "Portability of the GUI", *Byte*, 221-226, November 1991.

Appel, R., Tweede taal onderwijs en onderzoek naar tweede taal verwerving, in: M. Kienstra (Ed) *Referentieboek Onderwijs aan Anderstaligen*, Foris Publications, Dordrecht, 1989.

Bennik, F.D. and C.H. Frye, *PLANIT Author's Guide*, Systems Development Corporation, Santa Monica, 1970.

Bessagnet, M.N., T. Nodenot, G. Gouarderes, J.J. Rigal, "A New Approach: Courseware Engineering", in: A. McDougall and C. Dowling (Eds) *Computers in Education*, Elsevier Science Publishers B.V. (North-Holland), 339-344, 1990.

Boehm, B.W. "Software Engineering", *IEEE Trans. Comput.* C-25 (Dec 1976), 1226-1241, 1976.

Bork, A., *Learning with computers*, Digital Press, 1981.

Bork, A., "Producing CBL material at the Educational Technology Center", *Journal of Comp. Based Instr.*, Vol. 11, No. 3, 78-81, 1984.

Bork, A., *Personal computers for Education*, Harper and Row, 1985.

Bork, A., *Learning with Personal Computers*, Harper and Row, 1987.

Bork, A., Is Technology-Based Learning Effective?, *Contemporary Education*, Vol 63, No 1, 6-14, Fall 1991.

Bork, A., B. Ibrahim, B.Levrat, A. Milne, and R. Yoshii; "The Irvine-Geneva Course Development System", in: R.M. Aiken (Ed) *Information Processing 92* (IFIP '92), Elsevier Science Publishers B.V. (North-Holland), 253-261, 1992.

Bots, P.W.G., *An environment to support problem solving*, Ph.D. Thesis, Delft University of Technology, 1989.

Brussaard, B.K. and P.A. Tas, "Information and Organisation Policies in the Public Organisation", in: S.H. Lavington (Ed), *Information processing 80 (IFIP)*, North Holland Publishing Company, Amsterdam, 821-826, 1980.

Buve, R.W., *Specificatie en Realisatie van Educatieve Simulaties*, Afstudeerverslag Faculteit TWI, Technische Universiteit Delft, 1992.

Calvert, T. W., A. Bruderlin, S. Mah, T. Schiphorst, C. Welman, "The evolution of an interface for Choreographers", *Interact'93*, ACM/IFIP, 115-122, 1993.

Cerri, S.A. and J. Whiting (Eds.), *Learning Technology in the European Communities*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

Chambers, J. A. and J. W. Sprecher, "Computer Assisted Instruction: Current Trends and Critical Issues", *Comm. ACM*, Vol 23, No 6, 332-342, 1980.

Clanton, C., F. Iannella and E. Young, "Filmcraft in User Interface Design", *CHI92 Tutorial #9*, ACM, New York, 1992.

Coenen, M. and A. Vermeer, *Nederlandse woordenschat allochtone kinderen*, Uitgeverij Zwijsen, Tilburg, 1988.

COI, "Sunshine-project. 5. Rapport Specificatiefase: het weerstation", *PMI-reeks 28*, Project Management Infrastructuur INSP, COI, Enschede, 1987.

Curtis, B. and B. Hefley, "Defining a place for interface engineering", *IEEE Software*, March 1992, 84-86, 1992.

De Diana, I., *Het EDUC systeem: aspecten van een methodologie, ontwikkelingsmethode en instrumentatie voor tutorieel COO*, Ph.D. Thesis, Twente University, Enschede, 1988.

REFERENCES

DeMarco, T., *Structured Analysis and System Specification*, Prentice-Hall Inc., New Jersey, 1979.

Driessen, E.W., *De meerwaarde van een computer-ondersteund oefen- en simulatieprogramma*, Afstudeerverslag Universiteit van Amsterdam, 1993.

Eberts, R.E., and J. F. Brock, "Computer-Based Instruction", in *Handbook of Human-Computer Interaction*, M. Helander (Ed.), Elsevier Science Publishers B.V. (North Holland), 99-627, 1988.

ECC, *POCO Referentieboek*, ECC b.v., Enschede. 1989

Ellis, J., *Visible Fictions*, Routledge and Kegan Paul, London, 1982.

Elsom-Cook, M. T., "The ECAL teaching engine: pragmatic AI for Education", in: S.A. Cerri and J. Whiting (Eds.), *Learning Technology in the European Communities*, Kluwer Academic Publishers, Dordrecht The Netherlands, 329-340, 1992.

Fletcher, J.D., "The Effectiveness and Cost of Interactive Videodisc Instruction", *Machine-Mediated Learning*, Vol 3, No 4, 361-385, 1989.

Foley, J.D. & Wallace, V.L., "The Art of Natural Graphic Man-Machine Conversation", *Proceedings of the IEEE*, Vol 62, No 4, 462-471, 1974.

Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F., *Computer Graphics Second Edition,* Addison-Wesley, Reading, 1990.

Frenkel, K.A., "A Conversation with Brenda Laurel", *Interactions*, Vol 1, No 1, 44-53, 1994.

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

Gery, G., *Making CBT Happen*, Weingarten Publications, Boston, 1987.

Geuchies H.B. and P.M. Sleeuwenhoek, "Courseware Conversie van PLANIT naar C en EDUCALIB", *Intern rapport TWI*, Delft University of Technology, 1989.

Giddings, R. V., "Accommodating Uncertainty in Software Design", *Comm. ACM,* May 1984, Vol.27, No 5, 29-35, 1984.

Gimpel, J., *The Cathedral Builders*, Harper Perenial, 1992.

Godfrey D. and S. Sterling, *The Elements of CAL*, Reston Publishing Company, Reston Va, 1982.

Green, M., Report on Dialogue Specification Tools, in: G.E. Pfaff (ed.) *Proceedings of the Workshop on User Interface Management Systems*, Springer-Verlag, 9-20, 1985.

Grudin, J., S.F. Ehrlich, R. Shriner, "Positioning Human Factors in the User Interface Development Chain", in: J.M. Carroll and P.P. Tanner (eds) *Proceedings. CHI+GI87*, ACM, 125-131, 1987.

Grudin, J. and S. E. Poltrock, "User Interface Design in Large Corporations: co-ordination and communication across disciplines", *Proceedings CHI'89*, ACM New York, 197-203, 1989.

Harel, D., "Biting the silver bullet", *IEEE Computer*, January 1992, Vol 25, No 1, 8-20, 1992.

Hartemink, F.J.A. (Ed.), "Handleiding voor de Ontwikkeling van Educatieve Programmatuur Versie 3.0", *PMI-Reeks 12*, COI, Enschede, 1988.

Hatley, D.J. and I.A. Pirbhai, *Strategies for Real-time System Specification,* Dorset House, New York, 1988.

Hays, R. T. & M. J. Singer, *Simulation in Training System Design, Bridging the Gap between reality and training*, New York: Springer Verlag, 1988.

Hebenstreit, J., New Trends and Related Problems in Computer-Based Education, in: B. Gilchrist (ed.), *1977 IFIP Congress Proceedings*, North-Holland, 201-208, 1977.

Heckel, P., *The Elements of Friendly Software Design,* SYBEX, 1991.

Heines, J. M., *Screen Design Strategies for Computer Assisted Instruction*, Digital Press, 1984.

Helwig, E., "Functionele specificaties van script-editor (versie 1.10)", *PMI-Reeks No. 42*, Project management Infrastructuur INSP, COI, Enschede, 1988.

Henderson, A., A Development Perspective on Interface Design and Theory, in: J.M. Carroll (ed.), *Designing Interaction*, Cambridge University Press, 254-268, 1991.

Henkens, L. and K. van Deursen, "Courseware Development in the Netherlands", in: F. Lovis and E.D. Tagg (eds.) *Computers in Education,* Elsevier Science Publishers B.V. (North Holland), 559-563, 1988.

Hix, D. and H.R. Hartson, *Developing User Interfaces*, John Wiley, 1993.

Hoogeveen, *The Viability of Multimedia retrieval Systems for marketing and Sales*, Ph.D. Thesis, Delft

University of Technology, Delft, 1994.

Hooper, R., The National Development programme in Computer-Assisted Learning- Origins and Starting Point, *PLET*, Vol 11, No 2, 1974.

Ibrahim, B. et al., "Courseware CAD", in: *Computers in Education* A. McDougall and C. Dowling (Eds.), Elsevier Publishers (North Holland), 383-389, 1990.

Jackson, M., *Principles of Program Design*, Academic Press, New York, 1979.

Jacques, R., J. Preece, T.T. Carey, Engagement as a Design Concept for Multimedia, submitted to the *Canadian Journal of Educational Communication*, 1995.

JCAL, Special Issue Aspects of Courseware Authoring, *Journal of Computer Assisted Learning*, Vol 10, No 2, June 1994.

Jones, J. C., "How My Thoughts about Design have Changed During the Years", *Design Methods and Theories*, Vol. 11, No 1, 50-62), 1977.

Kearsley, G., "Authoring Systems in Computer Based Education", *Comm. ACM*, Vol 25, No 7, 427-437, 1982.

Kearsley, G., "Embedded Training: The New Look of Computer Based Instruction", *Machine-Mediated Learning*, Vol 1, No 3, 279-296, 1985.

Kearsley, G.P., and M.J.H. Hillelsohn, Human Factors Considerations for Computer-Based Training, *Journal of Computer Based Instruction*, Vol 8, No 4, 74-84, 1982.

Klep, J. and P. Kommers (ed.), *Courseware en Leerplanontwikkeling Didactische Systeemanalyse*, Instituut voor Leerplanontwikkeling, Enschede, 1989.

Kulik, J.A., Kulik, C.C. and Cohen, P.A., Effectiveness of computer-based college teaching: a meta-analysis of findings, *Review of Educational Research*, Vol 50, No 4, 525-544, 1980.

Kunneman, P. and M. Hertgers, "Computer based training in insurance companies and banks: a case of proven technology", in: J. Fricker (Ed) *Proceedings ETTE'92 Conference*, 63-71, Amsterdam, 1992.

Kurtz, B.L. and A. Bork, "An SADT Model for the Production of Computer Based learning Material", in: R. Lewis and D. Tagg (Ed.), *Computers in Education*, North Holland, Amsterdam, 375-384, 1981.

Kvavik, K.H., S. Karimi, A. Cypher and D.J. Mayhew, "User-Centered Processes and Evaluation in Product Development", *Interactions*, Vol 1, No 3, 65-71, 1994.

Kwantes, N.J.M. and J.G.L. Thijssen, *Indeling van leerdoelen binnen bedrijfsopleidingen, Wetenswaardigheden C 3.3*, Den Haag, ROI/CIVOB, 1986.

Kwantes, N.J.M., L.I.M. Strumpel and R. van der Weide, "Functie- en takenprofielen als opleidingsbasis", *Gids voor de opleidingspraktijk*, afl. 8 april 1991.

Laurel, B., *Computers as Theatre*, Addison-Wesley, 1991.

Laurel, B. (Ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley, 1992.

Leiblum, M., "A Media Selection Model Geared Towards CAL", *THE Journal*, Vol 7, No 2, 29-33, 1980.

Lesgold, A. M., "Critical Research Issues in the Implementation of the Next Generation of Educational Technology", *Machine-Mediated Learning*, Vol 1, No 4, 373-381, 1986.

Liker, J.K., M. Fleischer, M. Nagamachi & M.S. Zonnevylle, "Designers and their machines: CAD use and support in the USA and Japan", *Comm. ACM*, Feb. 1992, Vol. 35, No. 2, 77-95, 1992.

Mackay, W.E., "Managing CBI projects", *Eurit86*, Pergamon Press, 369-374, 1986.

Mantei, M.M., and T.J. Teorey, "Cost/Benefit Analysis for Incorporating Human Factors in the Software Lifecycle", *Comm. ACM*, Vol 31, No 4, 428-439, 1988.

McDermott Hannafin, K., "CBI authoring tools in Postsecondary institutions: a review and critical examination", *Computers Educ.* Vol 14, No 3, 197-204, 1990.

McKendree J. & J. Mateer, "Film Techniques Applied tot he Design and Use of Interfaces", in: B D Shriver (ed.) *Proceedings of the 24th Annual Hawaii International Conference on System Sciences (Software Technology Track)*, Hawaii, 32-41, 1991.

Merrill, M.D., "Learner Control in CBL", *Computer and education*, Vol 4, 77-95, 1980.

Merrill, M. D., E. W. Schneider and K. A. Fletcher, *TICCIT*, Englewood Cliffs, NJ: Educational

Technology Publications, 1980.

Merrill, M.D., "Component Design Theory: Instructional Design for Courseware Authoring", in: (Eds. J. Moonen and T. Plomp) *Eurit86*, Pergamon Press, Oxford, 213-223, 1986.

Ministry, *The Kingdom of the Netherlands facts and figures Education and Science*, Ministry of Foreign Affairs, 1991.

Moonen, J. C. M. M., *Computers veranderen de wereld, doch veranderen ze ook het onderwijs?*, Inaugurale rede, Universiteit Twente, Enschede, 1990.

Moonen, J. and C. van der Mast, "Production of courseware in the Netherlands", in: L.F. Lewis and B. Feinstein (Eds.) *Proceedings of the International Conference on courseware design and evaluation*, Ramat Gan, Israel Information Processing Association, 82-90, 1987.

Moonen, J. and J. Schoenmaker, "Evolution of Courseware Development Methodology", *Int. Journal of Ed. Research*, Vol 17, No 1, 109-121, 1992.

Moonen, J. and I. Stanchev, "Educational Interactive Systems Research: Instrumentation and Implementation", *Computers and Education*, Vol 21, No 1/2, 163-172, 1993.

Moran, T., "The Command Language Grammar: a representation for the user interface of interactive computer systems", *Int. Journal of Man-Machine Studies*, Vol 15, 3-50, 1981.

Myers, B.A., "Challenges of HCI Design and Implementation", *Interactions*, Vol 1, No 1,73-83, 1994.

NCP, *Succesfactoren bij het gebruik van educatieve programmatuur*, Nationaal Courseware Platform, Ministerie van Onderwijs en Wetenschappen, Zoetermeer, 1990.

NCP, *Handboek invoering en gebruik van educatieve programmatuur (3 Volumes: Beroepsonderwijs, Hoger Onderwijs en Bedrijfsopleidingen)*, Nationaal Courseware Platform, Ministerie van Onderwijs en Wetenschappen, Zoetermeer, 1991.

NCP, *Behind Courseware Development*, National Courseware Platform, Ministery of Science and Education of The Netherlands, Zoetermeer, 1992a.

NCP, *Eindrapport Follow Up Nationaal Courseware Platform*, Nationaal Courseware Platform, Ministerie van Onderwijs en Wetenschappen, Zoetermeer, 1992b.

Netherlands Design Institute, CD-Rom: Doors of Perception 1, Interactive Proceedings of Doors of Perception Conference (Amsterdam, 30-31 October 1993), *Mediamatic*, Vol 8, No 1, 1994.

Nielsen, J., "Paper versus Computer Implementations of Mock-up Scenarios for heuristic Evaluation", in: D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds.) *Human Computer Interaction Interact'90*, 315-319, Amsterdam: North Holland, 1990.

O'Malley, C., M. Baker and M. Elsom-Cook, "The design and evaluation of a multimedia authoring system", *Computers & Educ.* Vol. 17, No. 1, 40-60, 1991.

Orlansky, J. and J. String, J., "Cost Effectiveness of Maintenance Simulations for Military Training", *Machine-Mediated Learning*, Vol 1, No 1, 41-63, 1983.

Phillips, M.D., H.S. Bashinkski, H.L. Ammerman and C.M. Fligg, "A Task Analytic Approach to Dialogue Design", in M. Helander (Ed) *Handbook of Human-Computer Interaction*, Elsevier Science Publishers B.V. (North Holland), 835-857, 1988.

Plomp, T, K. van Deursen and J. Moonen (eds.), *CAL for Europe*, North Holland, Amsterdam, 1987.

POCO, *POCO referentieboek*, ECC Enschede, 1989.

POCO, *POCO Quality Assurance Plan Versie 2.0*, ECC Enschede, 1991.

POCO, *Eindverslag POCO project*, ECC Enschede, 1992.

PRINT, "Richtlijnen Courseware Projecten", *Rapport PRINT/OCW versie 2.0*, October 1990.

PRINT, "Achtergrondinformatie Onderwijskundig Ontwerp", *Rapport PRINT/OCW, versie 2.0*, January 1992a.

PRINT, "Technieken Courseware Ontwikkeling", *Rapport PRINT/OCW, versie 11.0*, April 1992b.

PRINT, "Courseware, Hoe maak je dat? Ervaringen met projectmatig werken", *Rapport PRINT/OCW, versie 2.0*, June 1993.

Richardson, J.J., "On-line Task Analyses in Maintenance Simulation", *Machine-Mediated Learning*, Vol 1, no.2, 153-173, 1984.

Rouse, W.B., *Strategies for innovation: Creating successful products, systems and organizations*, John Wiley, New York, 1992.

Rushby, N. J., E. B. James and J. S. A. Anderson, "A Three-dimensional View of Computer-Based Learning in Continental Europe", in: *Selected Readings in Computer-Based Learning*, N.J. Rushby (Ed.), Kogan Page, London, 1981.

Rushby, N.J., *Computer-based learning, State of the Art Report 11:4*, Pergamon, Maidenhead, 1983.

Rushby, N.J., "Perpetuating the Myth", in: *Learning Technology in the European Communities*, S.A. Cerri and J. Whiting, (Eds.), Kluwer Academic Publishers, Dordrecht The Netherlands, 511-517, 1992.

SCEN, "Advieslijst Keuzepakket Comenius 1992", *SCEN Special*, Andersen Consulting-ECC, Enschede, 1992.

Schoenmaker, J., "Linking New Applications to new design paradigms", *Computer and Education*, Vol 21, No 1/2, 181-192, 1993.

Schoenmaker, J., E. Nienhuis, J. Scholte and J. Titulaer, "A methodology for educational software engineering", in: *Computers in education*, A. McDougall and C. Dowling (eds.), Elsevier (North Holland), 189-194, 1990.

Schouw, C., *Onderzoek naar het ontwikkelen van multimediale CBT*, Doctoraalscriptie Faculteit der Sociale Wetenschappen, University of Leiden, 1992.

Seligmann, P.S., G.M. Wijers and H.G. Sol, "Analyzing the structure of IS methodologies, an alternative approach", in: R. Maes (ed) *Proceedings of the first Dutch Conference on Information Systems*, Amersfoort, 1989.

Sheingold, K. and M. Hadley, *Accomplished teachers, Center for Technology in Education*, Bank Street College of Education, New York, 1990.

Sherwood, B.A. and J.H. Larkin, "New Tools for Courseware Production", *Journal of Computing in Higher Education* Vol 1, No 1, 3-20, 1989.

Sherwood, B. A. and D. M. Andersen, "cT creates prize-winning portable physics programs", *Computers in Physics*, Vol 7, No 2, mar/apr 1993, 136-143, 1993.

Shlechter, T. M., "An Examination of the Research Evidence for Computer-Based Instruction", in: *Advances in Human-Computer Interaction* Volume 2, H.R. Hartson and D. Hix (Eds), Ablex Publishing Corporation, Norwood NJ, 1988.

Shneiderman, B., Direct manipulation: A Step Beyond Programming Languages, *IEEE Computer*, Vol 16, No 8, 57-69, 1983.

Shneiderman, B., *Designing the User Interface, Second Edition*, Addison-Wesley, 1992.

Simon, H.A., "The Structure of Ill-structured Problems", *Artificial Intelligence*, Vol 2, 181-200, 1973.

Smets, G.J.F., P.J. Stappers, K. Overbeke and C. van der Mast, Design in Virtual Reality: Implementing Perception-Action Coupling with Affordances, in: G. Singh, S.K. Feiner and D. Thalmann (Eds.) *Virtual Reality software and Technology*, World Scientific, Singapore, 97-110, 1994.

Sodoyer, B. R. and C. van der Mast, "Script-editor: een editor voor de specificatiefase bij het ontwikkelen van COO", *Rapport 87-68, Faculteit TWI*, Technische Universiteit Delft, 1987.

Sodoyer, B. R., "A tool to produce courseware specifications", in: *Computers in Education*, A. McDougall and C. Dowling (eds.), Elsevier (North Holland), 453-458, 1990.

Soerjadi, K.J., *From Script to Course: an Analysis of the Link Between the Design Phase and the Realization Phase*, Masters Thesis, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, 1990.

Sol, H.G., "Information Systems Development: A Problem Solving Approach", *INTEC*, Atlanta, 1988.

Sol, H.G., *Oratie Technische Bestuurskunde*, Delft University of Technology, 1992.

Soloway, E., M. Guzdial, K.E.Hay, Learner-Centered Design, Interactions, Vol 1, No 2, 36-48, 1994.

Spaans, A., J.J. De Graaff and C.A.P.G. Van der Mast, Automatische Generatie van Context-Gevoelige Helpinformatie uit een Kennis-Gebaseerde UIMS, AIgg Kennisgeving, Jaargang 7, No 1, 1-9, April 1994.

Stevens A. L., "The Next Generation of AI-Based Teaching Systems", *Machine-Mediated Learning*, Vol 1, No 4, 313-326, 1986.

Stoffers, S. and M. Terwindt, "Sunshine-project: 14. Specificaties van "Het Weerstation", Herziene versie volgens HOEP 3.0", *PMI-reeks No. 56*, COI, Enschede, 1988.

REFERENCES

Suppes, P. and E. Macken, "The historical path from research and development to operational use of CAI", *Educational Technology*, Vol 18, No. 4, 9-12, 1978.

Teleac, *PC-privé een cursus voor MS-DOS gebruikers*, Teleac, Utrecht, 1991.

Theunissen, F. (Ed.), *Werken aan Nederlands als Tweede Taal*, Katholiek Pedagogisch Centrum, Den Bosch, 1990.

Thijssen, J.G.L., *Bedrijfsopleidingen als Werkterrein*, Vuga, Den Haag, 1988.

Van Aalst, J.W., T.T. Carey, D.L. McKerlie, *Design Space Analysis as "Training Wheels" for User Interface Design*, to be published in the proceedings of the CHI'95 Conference, Denver, 1995a.

Van Aalst, J.W., C.A.P.G. van der Mast, and T.T. Carey, *An Interactive Multimedia Tutorial for User Interface Design*, submitted to CASE'95 Conference, Toronto, 1995b.

Van Beckum, J., M. Chanowski, F. Hartemink and J. Schoenmaker, *Sunshine-project. 1. Haalbaarheidsonderzoek en Opstartfase*, PMI-Reeks No. 21, COI, Enschede, 1987.

Van Beckum, J., "Management Issues for large scale Multimedia CBT projects", in: *Proceedings ETTE92 Conference* J.Fricker (Ed.), 181-187, Amsterdam 1992a.

Van Beckum, J., "Multimedia for flexibility in Training and Education: experiences from large scale multimedia projects", in: *TIME Europe 1992 Conference Proceedings*, EPCO International Eindhoven, 39-42, 1992b.

Van Beckum, J., "Toepassing van simulaties voor vliegtuigonderhoudstraining", in: *IT in beeld*, Stichting SURF, Utrecht, 37-40, 1992c.

Van den Berg, J. H., *Metabletica van de materie*, Uitgeverij G.F. Callenbach, Nijkerk, 1969.

Van den Camp, B.C.H. "Sunshine-project: 20. Externe evaluatie", *PMI-reeks No. 65*, COI, Enschede, 1988.

Van den Camp, B.C.H., W.L.M. van Montfoort and P.A.J. Perrels, "Van idee tot realisatie... Gids voor het specificeren van educatieve programmatuur", *PMI Reeks No. 54*, COI, Enschede, 1988.

Van der Mast, C., "A Modular CAI System", in: *Aspects of Educational Technology*, R. Budgett (Ed.), Vol 12, 336-344, Kogan Page, London, 1978 and in: *Selected Readings in Computer-Based Learning*, N. Rushby (Ed), 137-144, Kogan Page, London, 1981.

Van der Mast, C., "A portable program to present courseware on microcomputers", *Computers and Education*, Vol. 6, 39-44, 1982.

Van der Mast C., "Computer-based learning", in: *Computer-based learning, State of the Art Report*, NJ Rushby (Ed), Pergamon, Maidenhead, 1983, 111-124.

Van der Mast, C. en Chr. de Boer, "Educatieve Software op weg naar Volwassenheid", *Informatie en Informatiebeleid*, Vol 4, No 4, 47-54, 1986.

Van der Mast, C., "An AV-CAI-editor as a personal tool for teachers", in: *MICROS PLUS: Educational Peripherals* (S.Wills and R.Lewis Eds.), Elsevier Science Publishers (North Holland), 149-153, 1988.

Van der Mast, C.A.P.G. "Eindverslag van de projectmanager Cluster I Infrastructuur INSP", *PMI Reeks No. 67*, COI, Enschede, 1989.

Van der Mast, C., "Developing Courseware and Developing Highly Interactive Software", *Report 90-18, Faculty Technical Mathematics and Informatics*, Delft University of Technology, Delft,1990a.

Van der Mast, C., "On the Usability of Courseware: Looking for the Market", *Report 90-19, Faculty Technical Mathematics and Informatics*, Delft University of Technology, Delft, 1990b.

Van der Mast, C. "Naar methodisch ontwerpen van zeer interactieve programmatuur", in: M. Looijen and H.G. Sol (Eds.), *Computer en Educatie*, Kluwer Bedrijfswetenschappen, Deventer, 113-126, 1990c.

Van der Mast, C., F. Hartemink and L. Henkens, 'Case Study of a National Educational Software Development Program in the Netherlands', *Machine-Mediated Learning*, Vol. 3, 345-360, 1991.

Van der Mast, C.A.P.G., "Trends in Computer Assisted Instruction", in: *COO, State of the Art*, Hogeschool voor Economische Studies Rotterdam, 18-26, 1991.

Van der Mast, C. and J. Rantanen, "Next Generation Authoring Systems: integration of multiple methodologies and tools", in: S.A. Cerri & J. Whiting (Eds.) *Learning Technology in the European Communities*, Kluwer Academic Publishers, Dordrecht The Netherlands, 519-533, 1992.

Van der Mast, C. and Versendaal, J., "Separation of User Interface and Application with the Delft Direct Manipulation Manager (D2M2)", in: H.G. Stassen (Ed) *Analysis, Design and Evaluation of Man-Machine Systems* 1992, 163-168, Pergamon Press, Oxford, 1993.

Van Manen, H., *Tot u spreekt...: Hans van Manen, bijzonder hoogleraar*, Nederlands Instituut voor de Dans, Amsterdam, 1992.

Van Meel, J.W., *The Dynamics of Business Engineering*, Ph.D. thesis, Delft University of Technology, Delft, 1994.

Van Offenbeek, M.A.G., *Van methode naar scenario's: het afstemmen van situatie en aanpak bij de ontwikkeling van informatiesystemen*, Ph.D. Thesis, Vrije Universiteit Amsterdam, 1993.

Verhoeven, L. and A. Vermeer, "Ethnic group differences in children's oral proficiency of Dutch", in: G. Extra and T. Vallen (Eds.) *Ethnic minorities and Dutch as a second language*, Foris, Dordrecht, 105-131, 1985.

Versendaal, J. M., *Separation of the User Interface and Application, Ph.D. Thesis*, Delft University of Technology, 1991.

Vonk, G, "Computers in het Onderwijs", *Informatie en Informatiebeleid*, Vol 9, No 4, 31-35, 1991.

Wiechers, G., *Design and Implementation of a System for Computer Assisted Instruction*, Technical Report University of South Africa, 1975.

Wijers, G. M., *Modelling Support in Information Systems Development*, Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1991.

G. Wijnen, W. Renes en P. Storm, *Projectmatig werken*, Spectrum Markaboek, 1984/1988.

Wilson, J. and D. Rosenberg, "Rapid prototyping for UI design", in: M. Helander (ed.), *Handbook of Human-Computer Interaction*, Elsevier Science Publishers (North Holland), 1988.

Young, E. and C. Clanton, *Filmcraft in user interface design, Interchi'93 Tutorial #17*, ACM New York, 1993.

Yourdon, E., *Managing the Structured Techniques*, Yourdon Press, New York, 1979.

ABBREVIATIONS

AA	American Airlines
AMM	Aircraft Maintenance Manuals
ATA	Air Transport Association of America
BZD	Basisopleiding Zakelijke Dienstverlening
BMB	Baliemedewerkers Bedrijven
CAI	Computer Assisted Instruction
CAIASCO	CAI ASsembler COde
CAL	Computer Assisted/Aided Learning
CAT	Courseware Actoren Tabel (PRINT method)
CBT	Computer Based training
CHI	Computer-Human Interaction
CMI	Computer Managed Instruction
COLOS	COmputer Leerstof Oriëntatie Schema (PRINT method)
ECC	Educational Computing Consortium
ED	Educational Design (POCO method)
EDUC	Environment for Developing and Using Courseware
ERD	Entity Relation Diagram (POCO method)
ET	department Education and Training (case study 1)
FBS	Fixed Base Simulator (case study 2)
FD	Functional Design (POCO method)
FFD	Formal Functional Design (POCO method)
GUI	Graphical User Interface
HOEP	Handleiding voor het Ontwikkelen van Educatieve Programmatuur
IFD	Informal Functional Design (POCO method)
IMPS	Instruction Management Presentation System
INSP	INformatica StimuleringsPlan (1984-1988)
IS	Information System
LAT	Lestaak Analyse Tabel (PRINT method)
LOS	Leerstof Oriëntatie Schema (PRINT method)
MCSD	Modular CAI System Delft
MMI	Man-Machine Interface
MT	Maintenance Training
OCW	Ondersteuningsstructuur CourseWare ontwikkeling
OMT	Object Modelling Technique (POCO method)
OSM	Object System Model (POCO method)
PI	Programmed Instruction
PL	Project Leader
PM	Project Manager
POCO	ProgrammatuurOntwikkeling voor Computers in het Onderwijs
PRINT	PRoject Invoering Nieuwe Technologieën
QA	Quality Assurance
RS	Real System
SC	Steering Committee
SCCS	Souce Code Control System
SSOA	Semantic Syntactic model with Objects and Action (B. Shneiderman)

ABBREVIATIONS

SME	Subject Matter Expert
TAM	Taak Analyse Matrix
TD	Technical Design (POCO method)
TICCIT	Timeshared Interactive Computer Controlled Information TV
UI	User Interface

SAMENVATTING (Summary in Dutch)

Probleemstelling
Educatieve programmatuur of computer ondersteund onderwijs (COO) wordt gebruikt ter ondersteuning van onderwijs, training en individueel leren of als een vorm van educatief amusement. Het is moeilijk om de educatieve meerwaarde van COO boven traditionele onderwijsmiddelen aan te tonen. Toch neemt het gebruik toe bij vele soorten onderwijs en training. In dit proefschrift wordt ingegaan op de inrichting van het ontwikkelproces van COO.

Het ontwikkelen van COO is altijd onderkend als een moeizaam proces. Als een van de belangrijkste redenen daarvan wordt gezien, dat bij het ontwerp en de realisatie kennis en ervaring uit geheel verschillende disciplines betrokken zijn, zoals bijvoorbeeld kennis van de betreffende leerstof, onderwijskunde en didactiek, software engineering, kennis van verschillende media en communicatietechnieken en tenslotte project management. Tijdens de samenwerking van medewerkers van deze disciplines verloopt de onderlinge communicatie vaak moeilijk. Tengevolge hiervan wordt het estafettestokje - het ontwerpidee - niet goed doorgegeven. De kwaliteit van het oorspronkelijke ontwerpidee kan verloren gaan.

Het ontwikkelproces kan in de praktijk in enkele categorieën worden ingedeeld. In de *individuele* benadering wordt het gehele COO product ontworpen en gerealiseerd door één persoon, meestal een docent, voor gebruik door zijn/haar eigen studenten. Bij de *teambenadering* werken verschillende docenten samen, eventueel met een eigen specialisatie, zoals de betreffende leerstof, algemene of vakdidactiek en het programmeren. Bij de *projectbenadering* wordt hieraan de discipline van projectorganisatie toegevoegd. Bij de *industriële benadering* komt daar nog bij dat het product gemaakt wordt voor een educatieve markt, waardoor marketing activiteiten noodzakelijk zijn: de kwaliteit van een product is dan niet alléén doorslaggevend voor succes maar de keuze van de product/markt combinatie. In deze volgorde is bij elke benadering meer expertise en kwaliteitszin vereist. De eerste twee benaderingen zijn in dit onderzoek niet betrokken omdat zij niet geschikt worden geacht voor het ontwikkelen van strategische producten voor gebruik op grote schaal.

In dit onderzoek wordt een nieuwe theorie voorgesteld voor het inrichten van het ontwikkelproces van multimedia COO. Deze theorie is gebaseerd op de ervaringen die in de literatuur zijn gevonden, aangevuld met de resultaten van twee bijzondere projecten die nauwkeurig zijn geanalyseerd. De theorie wordt vervolgens getoetst aan de ontwikkelmethoden die bij drie COO projecten zijn gevolgd.

Resultaten van literatuuronderzoek en twee inductieve voorstudies
Acht in de literatuur of de praktijk gevonden methoden voor het ontwikkelen van COO zijn onderzocht op de wijze waarop zij invulling geven aan de wijzen van denken, modelleren, werken en beheersen; dit zijn vier aspecten van een bekend raamwerk voor het vergelijken van ontwikkelmethoden. De conclusie is dat sommige methoden veel aandacht vragen voor de kwaliteit van het team en een optimale samenwerking. Andere methoden besteden veel aandacht aan modelleertechnieken en ondersteunende gereedschappen. Ook besteden twee methoden aandacht aan specifieke instructiemodellen. Slechts drie methoden besteden uitgebreid aandacht aan de organisatie van het ontwikkelproces. Geen van de methoden biedt specifieke steun bij het ontwerpen en realiseren van de gebruikers interface en verschillende

media. Algemene conclusie is dat de aandacht bij geen enkele van de beschreven methoden evenwichtig wordt verdeeld over de hier genoemde wijzen van denken, modelleren, werken en beheersen.

Van zeven programmatuurgereedschappen die bepaalde fasen van het ontwikkelen kunnen ondersteunen wordt een overzicht gegeven. De commerciële gereedschappen hebben alle voorzieningen voor prototyping en media ontwerp maar bovenal ook voor de realisatie van het eindprodukt. De niet-commerciële zijn meer gericht op de vroege ontwerpfase, met name op het modelleren, en minder op de realisatiefase. Daarom werken de laatste een scheiding tussen ontwerpen en realiseren in de hand.

De eerste voorstudie betreft het modelproject *Sunshine* van het Informatica Stimulerings Plan INSP (1984-1988) van de Nederlandse Overheid. Dit modelproject liep tegelijk met het opstellen van een ontwikkelmethodiek voor COO ten behoeve van het INSP. Daarom waren er extra middelen beschikbaar om het ontwikkelproces tijdens het Sunshine project expliciet aandacht te geven en om voorbeelddocumentatie te maken.

De tweede voorstudie betreft het *PC-Privé* project van Teleac. Dit project is gekozen omdat in de literatuur genoemd wordt dat het ontwikkelen van actualiteitenprogramma's voor de TV en documentaire films als voorbeeld zou kunnen dienen voor het ontwikkelen van COO. Het *PC-Privé* project betreft de ontwikkeling van een multimedia cursus voor uitzending op de Nederlandse televisie. Deze cursus is enkele malen uitgezonden. Het onderwerp van *PC-Privé* is "MS DOS en de principes van de standaard toepassingen: tekstverwerken, desktop publishing, databanken, rekenvellen, boekhouden en communicatie".

De nieuwe theorie

De nieuwe theorie bestaat niet uit een duidelijke methode die eenduidig en letterlijk kan worden toegepast. De theorie wordt gepresenteerd als een samenhangend stelsel van een denkwijze, een modelleerwijze, een werkwijze, een beheerswijze en een ondersteuningswijze. Uitgaand van de benadering zoals gebruikelijk bij het ontwerpen van informatiesystemen, dienen de vragen *waartoe*, *wat* en *hoe* duidelijk en zo volledig mogelijk beantwoord te worden. Deze denkwijze leidt tot het onderscheiden van drie niveaus van modellering: het macro-, meso- en microniveau. Het *macroniveau* heeft betrekking op modellen die van belang zijn voor de beantwoording van de waartoevraag: de organisatorische contekst van het onderwijs of de training, de plaats in het curriculum, de problemen die bij de organisatie, het onderwijs of het leren optreden, de algemene leerdoelstellingen die geformuleerd kunnen worden, de eisen die aan een oplossing gesteld moeten worden, de criteria om vast te stellen of een oplossing aan de gestelde eisen voldoet en een analyse van de relevante eigenschappen van de gebruikers (studenten, docenten en anderen die een rol spelen bij de organisatie van het onderwijsproces). Het *mesoniveau* heeft betrekking op het beantwoorden van de watvraag: de leerstofinhoud en het netwerk van leerdoelstellingen voor de student, de rol van de docent, de instructiestrategie, de taakanalyse van het beoogde leerproces, het conceptueel ontwerp van de gebruikers interface en de mediaselectie. Het *microniveau* heeft betrekking op het beantwoorden van de hoevraag omtrent de interactie tijdens het individuele leerproces, met als belangrijkste onderdeel het stimuleren van motivatie en engagement bij de student. Het ontwerp van de gebruikers interface dient op dit niveau expliciet aandacht te krijgen.

De werkwijze wordt beschreven als een reeks processtappen waarbij voor elke stap wordt aangegeven welke disciplines en welke niveaus (macro-, meso- en micro-) daarbij betrokken dienen te zijn. In het bijzonder wordt daarbij uitgewerkt hoe de kwaliteitsbewaking voor de verschillende disciplines dient te worden georganiseerd.

Teneinde voldoende aandacht te besteden aan de media-aspecten tijdens het ontwikkelproces wordt het produceren van films, TV programma's en theateruitvoeringen als voorbeeld genomen. Film is een typisch "communicatie" product waarin het overbrengen van denkbeelden en emoties centraal staat. Het ontwikkelproces van een film is complex, kostbaar en uitermate multi-disciplinair. Voor de algehele productie van een film zijn niet alleen verschillende media-disciplines nodig (story schrijven, script schrijven, kamerawerk, acteerkunst, enz.), ook maken de financiering, de marketing en de distributie een essentieel onderdeel van de productie uit. Geconcludeerd wordt dat de rol van de producer of de regisseur zeer centraal staat in het "ontwikkelproces" van een film. Dit leidt tot de veronderstelling dat deze rol model kan staan voor de rol van de leider of regisseur van een ontwikkelproject voor COO.

De nieuwe theorie wordt samengevat in 8 onderdelen:
T1: Het ontwikkelen van COO moet gericht zijn op het oplossen van een "probleem". Een organisatie of persoon moet kunnen optreden als de "eigenaar" van dit probleem.
T2: De probleemeigenaar en de projectleider moeten aandacht besteden aan onderwijskundige, technologische en media aspecten van het COO product.
T3: Bij het modelleren van het probleem en mogelijke oplossingen moeten het macro, meso en micro niveau onderscheiden worden.
T4: Het specificeren van de gebruikersinterface dient op een professionele wijze plaats te vinden.
T5: Het ontwerp en de realisatie moeten worden uitgevoerd door verschillende teams.
T6: De projectleider moet de rol spelen van regisseur met een duidelijk beeld van het eindprodukt voor ogen. Hij moet alle benodigde disciplines kunnen overzien.
T7: Er moeten professionele hulpmiddelen voor kwaliteitsbewaking worden ingezet.
T8: Gereedschappen moeten ondersteunen dat het "estafettestokje" binnen het team kan worden doorgegeven.

Om de theorie te toetsen werden de volgende acht onderzoeksvragen in drie case studies beantwoord.
Hoe is tijdens het ontwikkelen de aandacht verdeeld tussen onderwijskundige en technologische aspecten?
Is COO gezien als een gereedschap en/of als een medium?
Zijn tijdens het ontwerp modellen gemaakt op het macro, meso en micro niveau?
In welke mate zijn verantwoordelijkheden gedefinieerd en doorgevoerd?
In welke mate is de communicatie tussen verschillende disciplines ondersteund?
In welke mate zijn ontwerp en realisatie gescheiden uitgevoerd?
Is het project geleid door één persoon met een duidelijke visie over het eindprodukt?
In welke mate zijn maatregelen genomen voor kwaliteitsbewaking?

Drie case studies
Er zijn drie verschillende COO-projecten geselecteerd. Het eerste project betreft het ontwikkelen van een cursus Basisopleiding Zakelijke Dienstverlening voor baliemedewerkers van de Rabobank. De cursus (het COO gedeelte) bestaat uit oefeningen en simulaties over o.a. betaaldiensten binnenland en buitenland, creditgelden, financiën en verzekeringen. Het project is binnen de bank uitgevoerd onder leiding van de afdeling Opleidingen Bedrijven. Voor onderdelen van het ontwerp en de realisatie werden soms externe partners ingehuurd. Bij dit project lag de nadruk op onderwijskundige en in veel mindere mate op technologische aspecten; COO werd gezien als een gereedschap, niet als een medium voor de student; uitgebreide macro- en mesomodellen werden gemaakt, maar micromodellen nauwelijks;

verantwoordelijkheden werden wel gedefinieerd maar niet volgens de bedoelingen nageleefd; communicatie tussen de disciplines werd niet expliciet ondersteund; ontwerp en realisatie werden niet volgens de theorie gescheiden uitgevoerd; de projectleider had weinig kennis van de leerstof; er werden geen maatregelen genomen voor kwaliteitsbewaking van het gehele ontwikkelproces. De gebruikte algemene methodiek voor projectmanagement was niet toereikend voor COO. Een product met weinig interactie was het resultaat, dat echter toch op grote schaal werd afgenomen door de deelnemende lokale banken

Het tweede project betreft een cursus voor onderhoudspersoneel van Fokker 100 vliegtuigen. De cursus (het COO gedeelte) bestaat uit oefeningen en simulaties over een vijftigtal onderhoudsprocedures aan het vliegtuig en de motoren, aan het elektrisch systeem en aan het avionicasysteem. De uitvoering van het project is door Fokker Aircraft uitbesteed aan BSO/Origin. Fokker heeft o.a. de leerstofspecialisten en de projectleider/regisseur ingebracht. Het project omvatte niet alleen het ontwikkelen van een COO-product maar tevens het inwerken van een klein team bij Fokker om het product volledig in eigen beheer te kunnen onderhouden. BSO/Origin heeft expliciet de opdracht gekregen om o.a. de kwaliteit van het ontwikkel*proces* te bewaken. Fokker zelf bewaakte de kwaliteit van het eind*produkt*. Bij dit project lag de nadruk zowel op onderwijskundige als op technologische aspecten; COO werd gezien als gereedschap, maar zeker ook als medium voor de student; modellen werden op alle drie de niveaus gemaakt; verantwoordelijkheden werden zorgvuldig gedefinieerd en dienovereenkomstig nageleefd; de communicatie tussen de disciplines werd met kracht ondersteund; ontwerp en realisatie werden gescheiden uitgevoerd; de projectleider had een duidelijke visie op vorm en inhoud van het product; er werden voortdurend maatregelen genomen en bijgesteld voor kwaliteitsbewaking van het gehele ontwikkelproces. De gebruikte methodiek voor projectmanagement was speciaal afgestemd op het ontwikkelen van COO. Na beëindiging van het project heeft Fokker Aircraft zelf enkele nieuwe versies van de cursus gemaakt. Het product wordt op grote schaal gebruikt.

Het derde project betreft de cursus Nederlands als tweede taal *Woordwijs*. Deze cursus is bestemd voor allochtone kinderen in de leeftijd van 8-12 jaar. Het COO product sluit aan bij de methode voor leesonderwijs *Wie dit leest* van Zwijsen. Doel is uitdrukkelijk alleen het *oefenen* van al bekende woorden in verschillende conteksten om de woordenschat van de kinderen uit te breiden. De cursus is ontwikkeld in het kader van het POCO-project van de Nederlandse overheid door Andersen-ECC in samenwerking met uitgeverij Zwijsen. Dit project heeft een bijzonder karakter door het deelnemen van partners met geheel verschillende belangen. Bij dit project lag de nadruk zowel op onderwijskundige als op technologische aspecten; COO werd gezien als gereedschap voor de leerkracht en als medium voor de leerling, meestal afkomstig uit een ethnische minderheid; modellen werden op alle niveaus gemaakt; verantwoordlijkheden werden gedefinieerd en dienovereenkomstig nageleefd; de communicatie tussen de disciplines werd ondersteund: deze was heel goed binnen het kleine ontwikkelteam waarvan een lid in verschillende disciplines ervaring had, maar wat moeizaam tussen het ontwikkelteam en het team van de uitgever door een verschillende achtergrond van de medewerkers; de projectleider was goed op de hoogte van de leerstof, de didactiek en informatietechnologie; er werden voortdurend maatregelen genomen en bijgesteld voor kwaliteitsbewaking van het gehele ontwikkelproces. Omdat het een product voor de onderwijsmarkt is en het gebruik door leerlingen zich over 5 jaar uitstrekt, moet worden afgewacht op welke schaal *Woordwijs* verkocht en gebruikt zal worden.

Conclusies

De antwoorden op de vragen hebben geleid tot het verwerpen van enkele onderdelen van de theorie. In de verrichte case studies bleek de rol van de probleembezitter van grote invloed op het verloop van het project. T1 kon niet worden verworpen. Het besteden van aandacht aan onderwijskundige, technologische en media aspecten bleek heel belangrijk maar niet voldoende voor het welslagen van COO. T2 werd daarom verworpen. Bij het modelleren bleek het onderscheiden van macro, meso en micro niveaus van groot belang T3 kon niet worden verworpen. Het op een professionele wijze specificeren van de gebruikersinterface bleek niet zo belangrijk als verondersteld. T4 kon daarom worden verworpen. Het uitvoeren van ontwerp en realisatie door verschillende teams (T5) kon niet worden verworpen. De regisseursrol van de projectleider bleek belangrijk. T6 kon niet worden verworpen. Het belang van professionele hulpmiddelen voor de kwaliteitscontrole was groot. T7 kon niet verworpen worden. De rol van gereedschappen was beperkt tot prototyping voor het bespreken en uitwerken van ontwerpideeën. T8 kon niet worden verworpen.

Uit de antwoorden op de vragen kan de conclusie worden getrokken dat de rol van de *regisseur* meestal wordt onderschat. Hij dient in staat te zijn de kwaliteitsbewaking van het *product* geheel voor zijn rekening te nemen. Ook kan de conclusie worden getrokken dat voor het ontwikkelen van COO een aangepaste methodiek nodig is. Een methodiek uit de onderwijskundige wereld ofwel uit de wereld van informatiesysteemontwerp of software engineering is onvoldoende. Onderdeel van de methodiek dient te zijn een expliciet quality assurance plan dat tijdens het project wordt uitgevoerd en eventueel aangepast. Een andere persoon of instantie dan de regisseur dient verantwoordelijk te worden gesteld voor de kwaliteitsbewaking van het ontwikkel*proces*. Voor een goed verloop van het proces is het nodig dat de teamleden ervaring hebben in meer dan één van de betrokken disciplines. Het naast elkaar zetten van personen met een achtergrond in verschillende disciplines is onvoldoende. Hun samenwerking, taakverdeling en onderlinge communicatie moet van te voren worden afgesproken. De beste integratie van expertise in verschillende disciplines wordt verkregen, wanneer een of meer teamleden substantiële ervaring hebben in meer dan één van de betrokken disciplines.

CURRICULUM VITAE

Charles van der Mast (1944) attended Canisius College in Nijmegen and Lyceum Immaculatae Conceptionis in Venray, and received his Gymnasium β diploma in 1963. He went on to study technical physics at Delft University of Technology. He graduated in 1971 with a masters thesis on patter recognition and neural networks using hybrid computing. From 1971 - 1972, during his military service, he worked at the RVO - TNO Physics Laboratories in The Hague on remote sensing using neural networks. In 1973 and 1974 he worked at the psychonomy department of the Institute for Perception TNO at Soesterberg on concept learning. This was his first experience in multi-disciplinary research outside engineering.

In 1975 he went back to Delft University of Technology, Faculty of Technical Mathematics and Informatics for a research project on Computer Assisted Instruction. He was one of the developers of the Modular CAI System Delft. In 1979 he initiated courses on developing educational software at Delft for informatics students. The goal of the courses was to teach technical students to develop interactive information systems using educational software as an example. At the same time he gave basic computer science courses.

From 1984 - 1988 he was part-time project manager for the Dutch National Information Technology Stimulation Program (INSP) at Twente University (Centre for Education and Information Technology, COI) and was involved with building an infrastructure for developing educational software. The HOEP methodology for developing educational software was developed under his supervision.

In 1987 he initiated courses on user interface design and extended his research into user interface design methodologies and tools. He was involved in the Delft Direct Manipulation Manager project (D2M2) and co-operated with the Faculty of Industrial Design Engineering in a virtual reality project Modelling Objects in a Virtual Environment (MOVE).

Since 1989 he has supervised many Delft students abroad during their master's project in the areas of educational software and highly interactive systems. From this experience a good international exchange network has been established with outstanding institutes and universities.

Currently he is a senior lecturer on the subject "computer and education" at Delft University of Technology.

AUTHOR INDEX

SUBJECT INDEX