CHAPTER 9

EVOLUTION OF COURSEWARE DEVELOPMENT METHODOLOGY: RECENT ISSUES

JEF MOONEN* and JAN SCHOENMAKER†

*University of Twente, PO Box 217, 7500 AE Enschede, The Netherlands
†Educational Computing Consortium, PO Box 217, 7500 AE Enschede, The Netherlands

Abstract

To improve the quality of courseware products and the efficiency of the courseware development process, a methodology based upon "courseware engineering", being a combination of instructional systems development and software engineering, has emerged over the last 10-15 years. Recently, software engineering aspects have gained extra attention. It is unclear to what extent a courseware engineering emphasis has contributed to an improvement of the quality of the courseware products or to the efficiency of the production process. However, continuing developments in hardware and software are opening new perspectives for courseware engineering and connected research.

Introduction

It has been claimed that the major future issue in relation to courseware development is to further stimulate the combination of instructional systems' development strategies and software engineering (De Diana, 1988). This combination should further contribute to the development of what De Diana calls "courseware engineering".

Instructional systems development is, as defined by Logan (1979), "A general systems approach with multiple components called phases which, operating amongst a certain sets of constraints, is used to produce an instructional system. The phases are sequential sets of activities called analysis, design, development, implementation and control." Engineering is, according to ANSI/IEEE Standard 729, "The systematic approach to the development, operation, maintenance and retirement of software".

In this chapter we will use "courseware development" to refer to the total activity that covers the whole process of creating a courseware product: the feasibility study, the design phase, the coding and production phase, and the distribution. By "courseware engineering" we mean the methodology that is the basis for the practice of courseware development up to the distribution phase. "Courseware" includes by definition a software product to be used for educational purposes, and may also include accompanying printed material or material using other media.
Improvement of the quality of courseware and of the efficiency of courseware development should be major goals for courseware engineering. It should be stressed, however, that although a courseware engineering approach might also contribute to the improved integration of courseware products into the curriculum, it will not solve in itself the problem of the integration of technology into the curriculum. The latter integration process contains a number of aspects that are difficult to formalize and therefore is less suitable than courseware development for a systems approach (Van den Akker, Keursten, & Plomp, this issue).

In this chapter we describe some historical developments in courseware engineering, illustrate these referring to courseware development experiments in the Netherlands and discuss the possible influence of developments in hardware and software on perspectives for the future. In addition, we describe research areas that could be explored in order to strengthen the development of the area of courseware engineering.

Considerations Related to Courseware Engineering

The application of instructional systems' development and software engineering principles to courseware development is not new. Bunderson (1970) published in that year an instructional systems development approach for the development of courseware. A good example of a product created with such an approach is the CREATE-curriculum, developed by Control Data Corporation on the PLATO system (Taylor, 1979). Bunderson (1981) also discussed the application of software engineering principles to courseware development in reference to the TICCIT system. The major problem he encountered was how to manage large-scale software development projects and, in particular, how to differentiate staff for the distinguishable phases that are used as a consequence of a systematic and structured approach. The books, Issues in Instructional Systems Development and Computer-Based Instruction, both edited by O'Neil (1979, 1981), give a good overview of the state of affairs by the late 1970s.

By the end of the 1970s, the process of developing courseware on mainframes had matured. The main characteristic of the development activities was a good balance of emphasis between instructional systems development and software engineering principles. The major problems at that time were related to management and organizational problems within the development process, to the distribution of the courseware and to the integration of courseware into educational practice.

The success of the microcomputer during the 1980s, however, created a regression in courseware development approaches and techniques: the individual approach regained popularity. Such an approach is based upon one individual who creates the design and the technical implementation of a program (Moonen, 1986). As part of this approach instructional issues get increased attention and the solving of technical problems in relation to the limited microcomputer environment also requires considerable effort and time. However, also because of this individual approach there is a lack of quality control of the programs developed, a lack of efficiency during the development process and many difficulties when programs have to be transferred to other computer systems and educational environments.

The efficiency of the courseware development process can be increased by following
an "industrial approach", in contrast to an individual or team approach (Moonen, 1987). An industrial approach implies that mass production techniques, incorporating project management and quality assurance approaches, should become the major characteristics for the development process.

Now, at the end of the 1980s and from a technical perspective, we seem to be back where the mainframe courseware development stopped at the end of the 1970s: microcomputers have become powerful enough and software tools are widely available to easily overcome technical development difficulties. Similar to the situation at the end of the 1970s, the main problems in courseware development are again related to managerial, integrational and organizational aspects.

Of course there are also other differences compared to the 1970s, for example:

- the scope of available types of courseware has broadened very much in the direction of exploratory learning and the use of software as a tool in the didactical process;
- software engineering principles have also matured and are even incorporated in software tools;
- much more practical usage experience is available and the awareness of implementation problems has grown considerably.

Nevertheless, and from an overall perspective, not so much progress has been established over the last 20 years: courseware development projects still run out of time, planned budgets are still being overrun, and available expertise to execute projects is still scarce.

Practical Examples

Circumstances in the Netherlands

To better perceive the practical state of the art relative to current practice in courseware engineering, this section will describe recent activities in the Netherlands. Why in the Netherlands? Because during the period 1982–1989 the Dutch government has created a situation that was, and still may be, unique. Since 1982 comprehensive plans for the introduction of computers have been executed in each educational sector in Dutch education. Aspects of the approach were approved by parliament over a number of occasions, so political and considerable financial support has been a constant and important factor. The "Informatics Stimulation Plan" (INSP), executed from 1984 to 1988, had a specific subtask which was the "construction of an appropriate infrastructure for courseware development and distribution, including the design of standards and developmental tools". For this specific activity a budget of 19 million US $ was available.

In addition, courseware development activities took place for primary, secondary and vocational education (Van Deursen, 1987, 1988). What has happened in the Netherlands during the past six years could present a good example of what can occur in reality when support is optimized.
A General Approach for Courseware Development

An important activity during the INSP was the exploration of different courseware development strategies in order to come up with a general methodology for courseware engineering. As a result, manuals for courseware development (Hartemink (Ed.), 1988; Van den Camp, Van Montfoort, & Perrels, 1988) were published. In these manuals the suggested approach followed the general guidelines of instructional systems developments and of software engineering, for example, using different distinguishable phases during the development process. Its main characteristic, however, was the explicit distinction between the design phase and the technical implementation phase.

**Design**
- Feasibility study
- Starting phase: planning, budget, quality assurance plan
- Specification phase: design, functional specifications for the program and the documentation

Result
A script and (possibly) a prototype on the design machine

**Technical Implementation**
- Global technical design: logical structure of the program
- Detailed technical design
- Programming and debugging
- Technical acceptance test

Result
A first working version of the software program on the target machine

**Try-out**
- Field testing
- Revision
- Global acceptance test

Result
The final version of the software program on the target machine

**Implementation**
- Completion of final version for distribution
- Development of support materials
- Distribution

**Maintenance**

Figure 9.1 Overview of the Dutch approach to courseware engineering.
These two phases were interconnected through a "script" which contained a complete and detailed description of the product to be developed.

**Comments on the Process**

As design and technical implementation were explicitly seen as separate activities, different hardware systems and development tools were considered for executing those phases. In this concept, the hardware to be used in the design phase should be much more powerful than the hardware necessary to run the final product. As a consequence, powerful design tools could be used to support the design phase, the development of a prototype and the iterative work to optimize the prototype. In the Dutch experience SUN-workstations, powerful PC-ATs and Macintoshs were used as design machines. A script-editor specifically developed for this purpose (Helwig, 1987), a Norwegian program called "Mosaikk" (In het Panhuis, 1987), and Hypercard were also used for the development of prototypes and interactive screen displays. A software development environment called "Software through Pictures" was used for the design of schemes, diagrams and the production of documentation. The method used was based on Yourdon structured design and adapted toward the needs of interactive software. Programming was mostly done in the C language, and to support these activities a specific library of tools was developed (Educalib; Van Wijk & Van Montfoort, 1988). Educalib had tools for screen layout, drawing pictures, printing (parts of the screen), response analysis, the manipulation of windows, and has a limited word processing facility.

The suggested methodological approach can be seen as a theoretical model and as a hypothesis of how courseware development should be executed in order to improve the efficiency and effectiveness of the development process. In order to test this hypothesis in practice, a specific project (the Sunshine project) was carried out following the suggested methodological approach as closely as possible. An evaluation of the project (Van den Camp, 1988), however, revealed the limited applicability of the methodology. The problem was not financial—considerable means were available—but one of project management and especially of a failure to realize smooth cooperation between the different actors in the development process. This partial failure was also due to unclear project goals and specifications and to the fact that systems designers and programmers did not keep to specifications. Compared to what was available as a general conclusion at the end of the 1970s with respect to courseware development, a remarkable similarity occurred. The lifetime of the INSP did not allow the initiation of a new project to test the hypotheses again. As a consequence of the results of the Sunshine experience, the INSP courseware development methodology was revised: project management aspects and quality control got more emphasis.

**Production Project**

The majority of activities in courseware development during the INSP involved the straightforward production of courseware. The major experience was that working with
great numbers of small developing groups almost always resulted in considerable time delays and budget problems, and as a consequence the planned final products could not be used on time by the target groups. To overcome these problems there was a shift during the INSP period from individual work toward more team-oriented work, basically in order to get more control possibilities.

Near the end of the INSP, and considering the lack of sufficient amounts of courseware in the Netherlands, the Dutch government decided to start two new major courseware development projects: one focused toward the development of courseware for higher vocational education, and one focused toward the development of courseware for the other educational sectors (Henkens & Van Deursen, 1988). The latter project was called the POCO Project (Moonen, 1988). A major characteristic of this project is that it follows in principle the methodological approach as described in the previous section, but with a stronger focus on project management and an even stronger distinction among the different phases within the courseware development cycle: (1) choosing priorities; (2) formulating product descriptions; (3) managing technical production; and (4) distributing the courseware. Different groups of people execute the different phases of each development cycle. The final result of each phase has therefore to be accepted and approved by the project management before the next phase can start. A detailed quality assurance plan steers the activities to be executed in this respect.

At the end of the first cycle of the POCO Project in December of 1988, a first set of courseware products should have been delivered. This was not the case. For most of the planned products time delays started to appear during the technical production phase. Lack of sufficient numbers of qualified personnel, elaborate contract negotiations between the many parties (authors of written material, software houses, publishers, copyright holders) involved in the project and time-consuming revisions because of faulty communications between parties can be identified as causes for these delays. As the time frame of the first cycle of the project was very tight (the project started in August 1987 and planned to deliver 18 courseware packages by the end of 1988) perhaps this planning was unrealistic. However, even with the very strong project management that was imposed upon this project and the very detailed quality assurance plan that was agreed upon, it was not possible to keep production on time.

**Conclusion from the Dutch Experience**

The conclusion of the Dutch experience so far is that, although great efforts have been invested to stimulate the creation of an applicable courseware development methodology, the result is not yet satisfactory. Characteristic of the Dutch approach was the strengthening of the effect of software engineering aspects on the courseware development cycle without neglecting the educational aspects. In practice, however, the projects keep running out of time. And even worse, because of time constraints and difficulties in the communication between software developers, educational specialists and teachers, the educational aspects do not get enough attention. Further experiences and research are therefore necessary.
Future Directions

Hardware and Software Advances

The necessary research, as indicated in the previous section, will certainly be influenced by new developments in hardware and software. New developments will create new possibilities for courseware engineering. The processing power and mass storage capabilities of new hardware open new dimensions. The NEXT computer, designed to be of major use in higher education, is a typical example of such developments (Webster, 1989). The growing flexibility of authoring environments (for example: Course of Action), the availability of manipulable environments such as Hypercard, the interest in multi-media approaches and easy access to multi-media databases (Fox, 1989) appear to again be stimulating the individual approach in courseware development. Software engineering, especially through CASE (Computer Aided Software Engineering; Gibson, 1989), is also getting much attention.

Reworking the Interrelationship between Individual and Structured Approaches

A relevant question for the future therefore is what should be the balance between individual approaches—which will be guided by educational, instructional and pedagogical considerations—and software engineering approaches, which will emphasize systematic, structured and project management aspects. Perhaps the individual approach should claim its main influence during the feasibility and design phase of a project and end with a prototype of the perceived product, while a team approach guided by software engineering principles should dominate a project as soon as production through technical implementation, try-out and practical implementation are carried out. During the try-out educational specialists and teachers must have opportunities to contribute to the testing. More importantly, educational specialists must be able to influence and guide the revision process, based on initial try-outs.

Research Directions

The experiences and development described in the previous sections help us to reflect on the three major problem areas that were identified earlier in connection with the establishment of a methodology for courseware development: the global organization of courseware development projects, the project management aspects, and the integration of newly developed courseware into a curriculum. The experiences also allow us to identify important questions and areas of exploration for future research in courseware engineering.

Organization and Management Aspects

Comparison with Other Branches of Software Development

Courseware development follows—to a certain degree—the rules of other branches
of software development. The common denominator in software and courseware development is the modelling of data, processes and interactions. In the past, software development in connection with administrative automation focused very much on modelling data and processes. In courseware development, however, the accent within the development process has been mostly focused on the interaction between the end-user (teacher or student) and the material to be learned. Because of the increasing complexity of both administrative software and courseware—the administrative software towards a need for more extended user-interfacing and better interaction with the end-user and the courseware towards separate modelling of data and processes—the commonalities between products in both cases are becoming more apparent and therefore it seems to be relevant to follow similar development methodologies in both cases. In other words, CASE approaches should influence courseware developments and courseware engineering should influence general software development. The question, however, remains if a systematic approach, as suggested by courseware engineering, optimizes the efficiency and effectiveness of courseware development. No hard evidence is available to support such a statement. Continuous research about specific variations of courseware development methodologies is therefore necessary. Such research should include aspects pertinent to the cost-effectiveness of courseware development (Levin, 1983).

Management of Large Courseware Development Projects

The next area of consideration relates to the management of large courseware development projects. The following questions arise from earlier experiences (Collis & Gore, 1987). How can large-scale courseware development projects be best managed (keeping the project on a fixed time schedule, budget control, quality assurance, efficient use of effective tools)? What are the requirements for appropriate and qualified personnel? How can personnel be differentiated among distinguishable phases? How can cooperation and smooth transition of results between groups working in different phases be optimized? What kind of knowledge should each group have about the expertise of the other group in order to facilitate mutual communication? The answers to such questions should be put together in a quality assurance plan that accompanies each courseware development project.

New Design and Development Environments

Another consideration is whether certain types of courseware applications can be modelled better with different development approaches. For instance, in modelling interactions it is quite common to make use of windows, icons, buttons, etc. A design method to model dynamic and complex interaction processes is therefore object-oriented. In contrast, modelling and implementing data is done by using relational database management systems. These fourth-generation software tools are very suitable for administrative automation. They are, however, also relevant for storing and
maintaining "educational data" like texts, pictures, formulas, etc. Is there one preferable courseware development methodology or should that choice be determined by the kind of application?

As a general trend, not only related to modelling, it becomes apparent that traditional authoring systems and authoring languages are evolving towards powerful and flexible design and development environments. Object-oriented approaches (Cox, 1987) and intelligent database management systems (Parsaye, Chignell, Khoshafian, & Wong, 1989) are getting much interest, particularly in relation to the development of multimedia courseware (Moonen, 1989). New computer systems such as the NEXT computer (Webster, 1989) include as a standard option a library of predefined objects to which some information has to be added in order to use or combine them within new applications. The question about the influence of objects on specific courseware development activities and courseware development methodologies in general is obvious. However, as it is clear that new directions in this area are steered by large industrial companies, there seems to be little opportunity for education to influence these developments. Should educational research in this area therefore primarily be focused on the identification of useful applications of available technologies?

Automation of Design Activities

The current high development costs of courseware and new developments as mentioned above will make it necessary to further rationalize the courseware production process. These circumstances will stimulate the development of generic elements—producing modules which can be re-used (application-independent modules like user-interfaces, data registration programs, install options) and producing half products (to be finished by a publisher and with options for adaptation of the product by teachers and, possibly, students). As industrial developers are already producing "objects" that can be used in a generic way and provide development tools to create new objects, should educational research be focused on the identification of which kind of "objects" are needed to support the development of courseware half-products?

Research Questions

Four main topics are covered by the considerations mentioned in this section: the efficiency and effectiveness of current courseware development methodologies, the need for one general methodology or several methodologies connected to specific kinds of courseware, the influence of new hardware and new software tools on existing methodologies, and how to support project management activities.

The following more specific research questions can be formulated in relation to these topics, summarizing the above discussion:

(1) Which phase, or combination of them, within existing courseware development methodologies, contributes most explicitly to the efficiency of the process, or to
the effectiveness of the product? What is the preferred expertise of the personnel working in those phases?

(2) What are the essential components of a quality assurance plan to support large-scale courseware development projects? What strategies should be followed in order to optimize the impact of such a plan on the development process?

(3) How will the availability of libraries of "objects", that can easily be put together to form new courseware products, influence the global courseware development methodology? Will the increasing power of new computer systems and the availability of user-friendly design and development tools make courseware development again become an individual activity? What will be the impact of such a development on the design and consequently on the quality of the courseware?

(4) What kind of objects do we want to have available in order to produce courseware half-products? What level of flexibility in courseware half-products can be considered as optimal in relation to their use by publishers and, later, teachers and students?

Integration of Courseware into the Curriculum

Value of Increasing the Adaptability of Courseware

A first consideration relative to the integration of courseware into the curriculum and teaching practice relates to the problem that many courseware products do not fit very well into the existing curriculum and practice (Plomp, 1988; Van Akker, Keursten, & Plomp, this issue). This is due to the fact that the courseware is not developed in conjunction with an existing curriculum, or because of all kinds of local circumstances, such as the way teachers prefer to utilize a certain teaching method, the emphasis teachers want to put on specific content aspects, the available hardware facilities, the time available for using the courseware, etc. A possible solution for these constraints could be to make it easier for the teacher to adapt available courseware products to his or her own needs. There are several technical approaches to allow the adaptation and change of specific parts (data) of a program, using simple editors. Install options (response time, number of exercises, difficulty level, teaching method, etc.) are other means a teacher can use to tune a program.

Influence of Formative Evaluation During the Development Process

A second consideration refers to the fact that describing specifications of a product beforehand is very difficult, especially with respect to the interaction of the user with the program. A prototype of the product almost always elicits comments and suggestions for alterations. Formative evaluation is therefore an absolute necessity in courseware development (Kurland, 1989; Moonen & Schoenmaker, 1986; Burkhardt, this issue). However, due to time and financial constraints within projects, one can have doubts about how seriously formative evaluation is pursued in practice (Moonen, 1989).
Importance of a Model of the User

As a final consideration, we have to remember that when building a courseware product, one uses implicitly a "model of behavior of the end-user—teacher and student". Assessing the impact of such courseware should therefore focus on this implicit model. However, assessment is done very often independently of and without explicit reference to such a model (White, 1989). To enlarge the likelihood that the impact of the use of courseware will be discovered, assessment suggestions should be integrated into the courseware itself. This approach could be especially relevant for new forms of courseware, for instance in the case of exploratory learning.

Research Questions

Three main topics are covered by the considerations mentioned above: use of teacher toolkits, use of formative evaluation, and integration of assessment methods in the courseware. Research questions about these topics can be formulated as follows:

1. What kind of tools should be made available to teachers in order to make it easy for them to adapt courseware to their own teaching styles and preferred didactical approaches?
2. What levels of flexibility are appropriate in relation to the different kinds of courseware and to the different kinds of educational environments in which the courseware can be used?
3. How can ongoing formative evaluation be organized and integrated into the courseware development process?
4. What kind of strategies can be followed to add assessment suggestions into the courseware itself?

The overall research question is how the topics mentioned above, separately or combined, can improve the integration of courseware into the curriculum.

Conclusions

In this chapter we have suggested that courseware engineering should integrate the relevant components of software engineering techniques and instructional systems' development aspects. Perhaps the emphasis over the last years has been focused too exclusively toward the software engineering aspects. Because of the increasing power of modern computer systems and the practical difficulties in dealing with users—certainly in education—as well as with curriculum and cultural differences between many target groups, there is the danger that this emphasis will remain concentrated on technology-based developments, and particularly be directed toward an increase in the efficiency of development work. Developments at the European scale, for instance in projects such as DELTA, clearly indicate such a trend. The "quality of courseware" problem and the "integration of the courseware" in the daily practice of
education could, therefore, once again, be postponed "to be solved later". This should not happen.

On the other hand, new developments in hardware and software will increase the possibilities for individuals to become once again, as in the 1970s, deeply involved in the courseware development process. Such an approach will increase the likelihood that courseware will fit particular curricula and pedagogical concepts. On the other hand the distribution of such courseware and possibilities for adaptation of courseware by others to fit their instructional approaches will not be facilitated. As a result, much individual effort will be invested in a repetitive fashion. One can question the efficiency and cost-effectiveness of the development process in such a situation.

A combined strategy in which an individual approach, steered by educational technologists using new courseware development tools, will deliver a prototype of the product, followed by an industrial approach mainly based upon software engineering principles, is therefore recommended. Research should find out what should be the balance between the two components in such a strategy.

References


