Journal of Computer Assisted Learning (1998) **14**, 19–30

A method and tool for the design of educational multimedia material

E.J.R. Koper
Centre for Educational Technology, Open University of the Netherlands

Abstract This paper deals with the problem of the specification of the pedagogic design of interactive multimedia and specific telematics applications developed for use in education. A method to specify the design and a supporting tool (*SHOC-TOOL*) are described. The idea is that the use of such a method and tool can make the development process more efficient and can lead to a more professional approach to the pedagogic design of educational software in general. Resulting design specifications can be assessed for completeness, allow for comparisons among designs and are relatively easy to read for others than the designer himself. The method and the tool both stimulate the re-use of existing design and code segments. *SHOC-TOOL* is described along with several examples from real practice. The paper concludes with a discussion of the experience of its introduction and use.

Keywords: Multimedia; Object-oriented design; Pedagogic design

Introduction

One of the problems in developing educational software, whether it concerns interactive multimedia or specific telematics applications, is the issue of the pedagogic design of the programs and the methods and tools to support the pedagogic specification process. This is especially true in situations where different kinds of high quality computer programs are to be developed by professional multi-disciplinary teams. At the moment the available methods and tools for the specification of the pedagogic design are not in balance with the rich choices of methods and instruments which are available at the technical level. Work in pedagogic design specifications has been done in the field of intelligent tutoring systems (Livergood, 1990-1991), in the field of hypertext design (Poncelet & Proctor, 1993; Cates, 1994) and in the field of automated instructional design (Pirolli & Russell, 1990; Spector *et al.*, 1993). Most of the methods described are only suitable for the design of specific types of applications and not for a broad range of applications, varying from simple

Accepted: 15 May 1997

Correspondence: E.J.R. Koper, Centre for Educational Technology,
Open University of the Netherlands, Valkenburgerweg 167, 6419 AT Heelen, The Netherlands.
Email: rob.koper@ouh.nl

tutorials to multiuser and multimedia collaboration and simulation programs. It seems that in most projects pedagogical design is specified with rather *ad hoc* methods. This could have consequences on the pedagogic quality of the resulting product and on the efficiency of the development process.

On the quality side, designers should have access to comparable design specifications of existing programs and to the experience gained with the use of these programs, not only to learn from, but also to create the possibility to re-use elements of existing designs. When the format of the design specifications are *ad hoc*, comparisons are difficult to make and there is no possibility for a systematic culmination of knowledge within a designers' community. Another quality-related problem is to be found in the communication of a pedagogic design within a development team: a design must be clear and easy formatted in order for team members to assess the design validity.

On the efficiency side, the problem with *ad hoc* design specification methods is that they do not stimulate the systematic re-use of existing design elements and existing program code. At this moment most computer programmers use object-oriented design and programming tools which are optimised for re-usability. When an instructional designer has no way to access the objects and classes which are already in the libraries, the re-use of existing code will be restricted. One way forward is to give the designers access to existing object and class libraries. The problem however is that these tend to be rather technical, extensive and complex for an instructional designer. The library should in fact be made visible in a way that designers can understand easily. Another efficiency problem with using *ad hoc* design specification methods is of assessing the completeness of the design. One of the biggest problems for an efficient development process is that designs are (too) incomplete from the start. This is even a problem for those who use rapid prototyping as a development method, because it results in many costly development cycles.

To deal with these problems, a project at the Dutch Open University aims to develop a pedagogic design specification method for educational software which is not bound to specific instructional design principles or specific principals from learning theories. This means that the method must allow designs to be based on different theoretical perspectives.

The method is integrated into 'PROFIL' (Koper, 1991; 1992a; 1995) which has been used for the design and production of several dozens of high quality educational computer programs in the past years. To support the specification of the pedagogic design a tool, *SHOC-TOOL* (Koper & Slootmaker, 1993), has been developed. In this paper the specification method is described using *SHOC-TOOL* although the method is not bound to the use of that tool and it can be used with other tools (for example, text processing and drawing tools). At the end of the paper several examples are presented. First, however, the structure of pedagogic designs will be discussed further.

Structuring the pedagogic design

The pedagogic design specification method developed is based on some of the principles of 'object-oriented design' as it is known in the field of software engineering. However, in the software engineering literature this concept is not very strictly defined. Sommerville (1985, p. 68) distinguishes three types of design methods: top-down structured design; data-driven design and object-oriented design. Top-down structured design is based on the work of Wirth (1986) and Dijkstra (1984) with algorithmic decomposition to break down a complex system into subsystems. Data-driven design (Jackson, 1983) is based on the idea of mapping inputs and outputs of a system. Object-oriented design (Meyer, 1988; Blair *et al.*, 1991; Booch, 1994) is based on the idea of modelling a system as a collection of co-operating objects in which individual objects are treated as instances of a class, within a hierarchy of classes. Several methods and tools have been developed to specify a system in object-oriented terms. Some of them are formalised in such a way that a prototype can be derived automatically from the design specifications (Jackson, 1995).

When applying these object-oriented concepts and methods to the field of pedagogic design (Koper, 1992b), adaptations have to be made. This is mainly due to the fact that the concepts are too strictly bound to software development. The underlying assumption in the approach discussed in this paper is that the benefits of using an object-oriented approach in the field of pedagogic design is not limited to a specific medium. For a more detailed description of the design approach, see Koper (1995).

The basic concept is that of a 'pedagogic scenario'. This is a model of an encapsulated time-space relationship, in which an actor interacts with objects in order to obtain certain learning outcomes. The object-oriented design process is seen as one of designing pedagogic scenarios.

The time-space relationship of a pedagogic scenario can be defined in different ways: more space-related, more time (or 'process') -related or somewhere in between. This depends on the problem at hand. A good metaphor — especially for the more space-related scenarios — is to see the scenarios as 'virtual learning environments' or 'virtual rooms' in which students interact with objects. The space metaphor is used however for time-related scenarios and in that case the 'rooms' must be given names of the central activity or the output generated by the activities. The metaphor 'room' must not be taken too literally: also outside environments or imaginative environments may be included. Four different classes of objects in rooms are distinguished:

- communication objects,
- tools,
- background objects,
- connection objects.

Communication objects are the people with whom the student interacts, like teachers, other students and experts. *Tools* are all the (real or virtual) non-communication objects that can be manipulated by the student or by other communication objects in the room. *Background objects* are (real or virtual) objects which cannot be manipulated, but which set the context. *Connection objects* are

the (real or virtual) objects which are needed to go from one room to the other, e.g. doors and signs.

Besides the classes of objects, another distinction is made according to the status of an object in the design. There is a distinction between: functional objects, mediated objects and standard objects.

Functional objects are objects which are considered as possible objects in the scenario by the designer because of their pedagogic functionality and not (yet) by the medium in which it is implemented. For instance, a designer thinks of a book a student has to read in order to learn some facts. The functionality is what matters here: later on the designer can decide to put the facts into an object in the computer program or decide to print a real text book, depending on factors such as costs, access, feasibility and flexibility. When designing a computer program, the selection of the computer as the primary medium is already made. It is however well worth the effort of reconsidering the implementation of functional objects in one of the other available media when the design is complete. For instance, at the end it may turn out that it is better to implement certain elements of a design in printed text instead of in the computer program. One way of viewing functional objects is as a set of functions grouped in natural units, providing a higher level of abstraction in the design process than the usual approach of listing the separate functions.

Mediated objects are objects in the design which are decided (controlled) by the medium in which they have to be implemented.

Standard objects are specially created class objects of which different specific coded objects can be derived by instantiating certain variables. They can be used in different designs and can be adapted to a certain extent to fulfil the specific needs of the designer. Standard objects are the basic elements for the re-use of existing code in a design. Another form of re-use also exists: a designer can specify an object and save it; a programmer creates it when implementing the design. At a later stage the same design object can be used in a new design: the designer having the knowledge that the code already exists. The problem with standard objects is that the code of the objects is not optimised for re-use and it will probably take more effort to adapt the code to specific needs. When a certain object is re-used more than once, it can be transformed into a standard object.

Apart from a distinction in classes and in the design status of objects, a distinction is also made in the view of the objects. So, there are four possible views of objects (depending on the kind of object):

- a pedagogic view,
- a display view,
- a content-related view,
- a technical view.

A good design and development environment must make it possible to allow for all these views on the same objects, so that an instructional designer is able to view the object from the pedagogic point of view, a content matter specialist can look at the content of objects, a graphical designer (or audio-visual producer) can look at the display qualities and a programmer can look at the code of the object. Not all objects have to support all views however: some objects for instance do not contain content or are not displayed on the screen.

SHOC-TOOL: a description

SHOC-TOOL is a tool to support the instructional designer to specify the pedagogic design of an educational application. This support is three-fold:

- it gives information about the design process and the steps to be taken;
- it contains editors and tools for the different components of the design;
- it has libraries of standard objects.

The output of *SHOC-TOOL* is a complete specification of the pedagogic design of the program to be developed. *SHOC-TOOL* supports the designer in his task, but has no build in intelligence: it is not an expert system nor a decision support system. The pedagogic design process as described in the PROFIL methodology comprises five steps all supported by *SHOC-TOOL*:

Step 1 An analysis of the objectives which are to be achieved by the software and an analysis of the target group characteristics in terms of prior knowledge, learning needs and situational circumstances.

Step 2 An analysis of the pre-conditions for the development of the software: in which setting is it to be used? What are the possibilities of the target computer in terms of the communication codes supported (audio, text, video, telecommunication)? Which are the other available media in the educational setting besides computer programs (teachers, books, audio visual aids, etc.)? What delivery medium will be used: CD-ROM, Internet WWW or FTP, diskettes?

Step 3 A selection must be made of a pedagogic model which is a set of theoretical constructions dealing with (aspects of) learning, media and instruction. They are used as leading principles in the design, e.g. the principles of collaborative learning (Scardamalia & Bereiter, 1993/1994; Watabe *et al.*, 1995) or experiential learning (Kolb, 1984). A collection of models is sampled and described by Kearsley (1995).

Step 4 The design of the pedagogic scenario.

Step 5 The selection of the media in which the objects will be implemented and the selection of communication codes for all functional objects.

For management purposes a 6th step is included in *SHOC-TOOL*: the justification of the design.

SHOC-TOOL itself is also build with the same design method as is described here and is implemented with a 'space metaphor' in an abstracted OU standard screen representation with seven rooms.

The entry room

After starting the program the designer will come into the 'entry room' where he meets a porter who explains the objectives of *SHOC-TOOL* and gives navigational advice. The designer may ask (pre-programmed) questions to the porter about the program and the navigation. In the entry room there is also a map to show the structure of the program and the advised path through the rooms. The designer can go from one room to another by means of 'doors' which are visible in every room. A sign (arrow) points at the door in the advised path.

The data room

The next room the designer will visit is the 'data room' in which he can enter the data of Steps 1 and 2. In this room the designer will also meet the 'coach' which is present from now on throughout the program in every room. The coach gives instructions to the designer and answers content-related questions (how to specify objectives, etc.).

The model room

Next, the designer will go to the 'model' room in which the selection of a pedagogic model (Step 3) is made. The question which must be answered by the designer is: 'Which pedagogic model is the most suitable for attaining the specified objectives with the specified target group'. The designer can decide to select a standard model here, or to define his own model. The model is presented in text format, giving information about the following topics: general description of the model, the learning objectives it is suitable for, the target groups which it is directed at, the use of media, the underlying learning and instruction theory, empirical research which is available, examples and literature. When a designer chooses to specify a pedagogic model, he must sample the above information as much as possible himself.

The design room

The next room is the 'design room' in which the designer is asked to specify the pedagogic scenario (Step 4). The pedagogic scenario is directed at the learning objectives and target group characteristics and is inspired by the principles of the pedagogic model. This is a creative process: *SHOC-TOOL* provides the opportunity to structure and specify the resulting design.

The first step for a designer is to create a map of rooms and a preferred route through the rooms. The map can be considered as a chain of relatively independent learning environments. Every room stands for a demarcated set of learning activities the students have to undertake in order to attain certain (sub-)learning objectives. The design must be set up so that the completion of the total set of activities in the different rooms leads to the intended learning objectives. The designer can edit the map in the design room: he can select a standard map (which can be adapted when needed), select a previously saved map or make a new one. In the latter case he must create the rooms of the design, give each of them a name and specify the function of the different rooms. Also at this level a designer can chose between new rooms and standard rooms. With the editor the designer can specify the relationship between the rooms — whether a certain room can be entered from a specific room — and the preferred path through the rooms. An example of a designed map of a program in environmental law (Daal *et al.*, 1996) is given in Fig. 1. The map of the pedagogic scenario is to the right and shows four rooms: an entry room, an intake room, a study room and a test room. When the designer clicks on a room he sees a menu with choices (name, function, text colour, accessibility, connection, route, arrangement, copy, save to list, print, delete). With the option 'function' the designer will get a text editor in which he is asked to specify the function of the room.

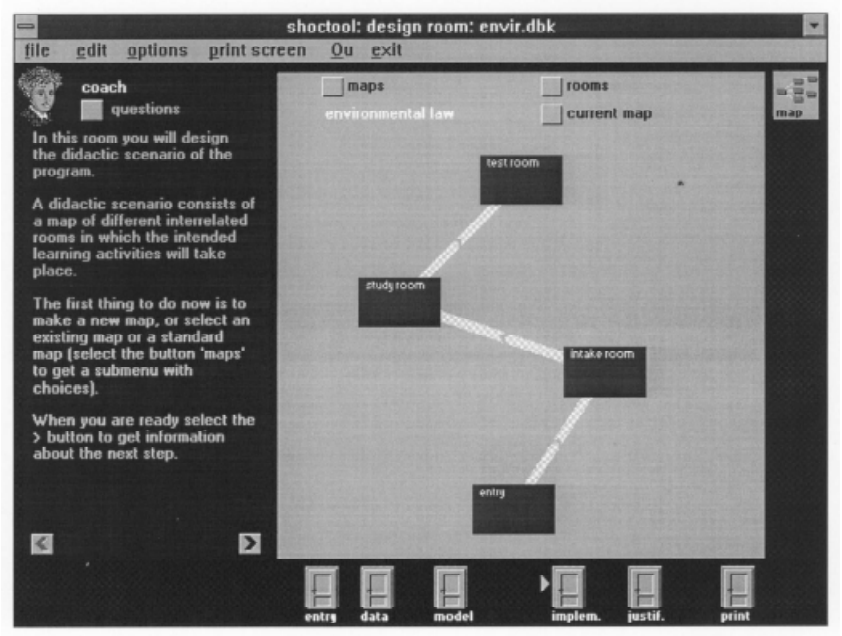

Fig. 1 An example of a map designed with *SHOC-TOOL*

To give an impression of the design in the example: the major function of the 'entry room' is to give the designer information about the objectives of program and to handle the student file (on diskette). In the 'intake room' the student's background (most are adult students) and study needs are assessed to create an adapted study plan. In the 'study room' the actual learning will take place. The materials, exercises and cases on environmental law are selected according to the study plan provided in the intake room. Not only the elements which are in the computer program are present in the room, but all the objects needed to complete the task. In the case of the study room there will be a printed text book which contains most of the texts to be studied and a computer program which provides the exercises, cases, feedback and the study plan. In the 'test room' there is an assessment of the knowledge the student has acquired on the topics specified in the study plan.

With the option 'arrangement' the designer will 'enter' the room, so he can view or edit the arrangement of the room. Figure 2 gives an example of the arrangement of the 'intake room' of the environmental law program.

When the intake room is entered for the first time it is empty. The standard question the designer must ask himself in this situation is: what activities is the student to perform in this room in order to fulfil the needed functionality (the design is centred around the student activities) and what objects students need to interact with. The editor in the room permits the addition of three types of object — communication objects, tools and background objects — by selecting buttons. Pressing one of the object buttons results in a menu with choices (new, select, standard, print, delete, import, export, paste). With these options it is, amongst others, possible to create a new object, select one of the objects which

are saved in previous designs by the designer or to select adaptable standard objects. The first thing the designer has to do when he enters an empty room is to put the communication object 'student' somewhere in the room. When the designer clicks on an object, such as 'student', he gets a menu with the choices: name, function, text colour, rotate, copy, save to list, print, delete, implementation. The functionality of every object has to be described ('function'). For the object 'student' this means that the activities of the student in relation to the other objects in the room are described. Other objects can be added when needed. In the example two standard objects have been added (the name starts with 's:'), namely the 'information tool' and the 'intaker'. The first one is an object present in most of the OU-programs. It gives general information about the program (objectives, target group, navigation, map, credits, etc.) to the student. The second one is a special case of the general 'communication class object' used for a lot of Open University programs.

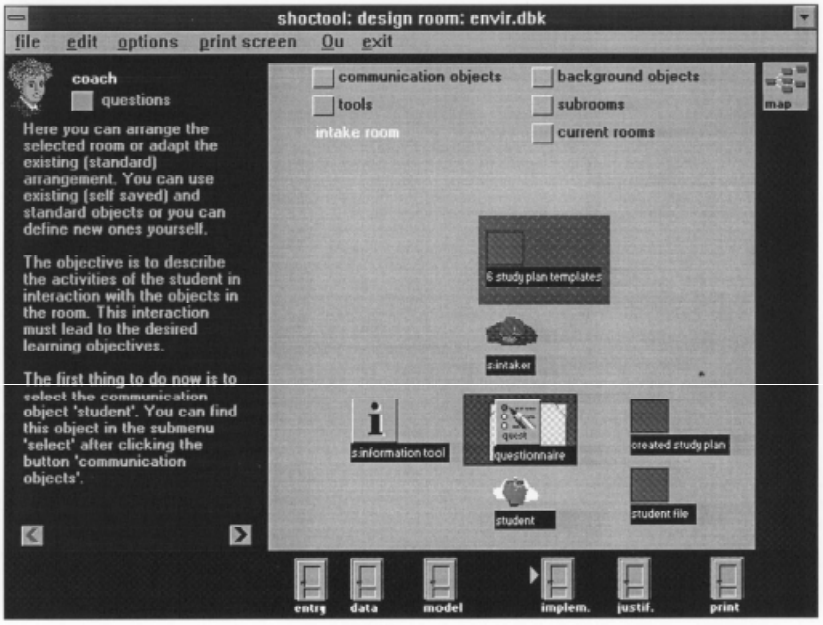

Fig. 2. Arrangement of the intake room

The functional description of a standard object is fixed, but there is an extra edit field where the desired adaptations can be specified. When every function of every object is defined, the implementation of the different objects has to be decided upon. The designer selects the menu-option 'implementation' for every object and gets a tool with two list boxes: one for the selection of the physical media the object is to be implemented in (real persons, print, computer program, audio tape, etc.) and one for the selection of the media codes for the implementation (audio, text, animation, full screen video, etc.). In an edit box the designer can give comments on the implementation. The designer can also go directly to an editor in which he can make a small prototype or graphical representation of the object.

The implementation room
When the implementation of objects is decided, the designer can go to the implementation room for a tool which gives an overview of all the objects and all the implementations in the design (Step 5). The objects in the design can be sorted on media, type of object or room. When sorted on media one gets an overview of all the objects which must be developed for the different media: the computer program, printed text, audio-visual materials and the real persons (e.g. tutors) needed for the study module. In this room the designer can also reconsider the implementation of objects and make changes where appropriate.

The justification room
The justification room provides a text editor in which the designer can type his arguments on why he has set up the design in the given way (step 6). What options are considered? Why are the selected media chosen (functionality, feasibility, accessibility, costs?). This justification follows the steps described in Koper (1989).

The print room
The print room provides various options to print the design. It is possible to make a full print of the whole design, including pictures, or just elements of the design.

Examples: simulations of authentic environments

Pedagogic scenarios which have attracted much attention recently are based on constructivistic approaches (Brown *et al.*, 1989; Duffy & Joanassen, 1991; Choi & Hannafin, 1995). The basis is that knowledge is bound to the situation in which it is learned. In order to learn, students must act in environments which replicate the real expert environments as much as possible (so called 'authentic' environments). Guidance at the context level can be given by experts performing the activities, leaving the students in the role of a 'legitimate peripheral participant' as Lave and Wenger (1991) would call it. Another possibility is to let students interact with each other in the environment according to collaborative learning principles (e.g. Dillenbourg & Schneider, 1995).

SHOC-TOOL is also suitable to design applications in this area, because it centres around the activities of the learner. Rooms can be designed in such a way that they represent a simulation of an authentic environment. An example in the field of 'leadership styles' will be dealt with to illustrate this.
This program (Hoogveld *et al.*, 1996) — which will be a part of a OU social sciences course on 'leadership' — gives the student the opportunity to test his own leadership style in two ways: by filling in a test about leadership and by confronting the student with his own leadership style in practice. In the latter case the student will act as a temporary manager in a postal firm, called *Fastgo*. Sitting in his office he can decide to walk around, ask questions, or just sit and wait. Whatever he chooses to do, he will be confronted with different problems, such as disturbances in the sorting process and problems between employees.

In every situation the student can decide what to do. When he makes a decision this is scored as a kind of leadership style. Later on his performance is confronted with the test results. In this way he will not only learn the relationship between formal tests and actual practice, but also gets some leadership experience learned in a simulated authentic environment. The *SHOC-TOOL* design comprises two maps: one of the program structure and one of *Fastgo* (Fig. 3).

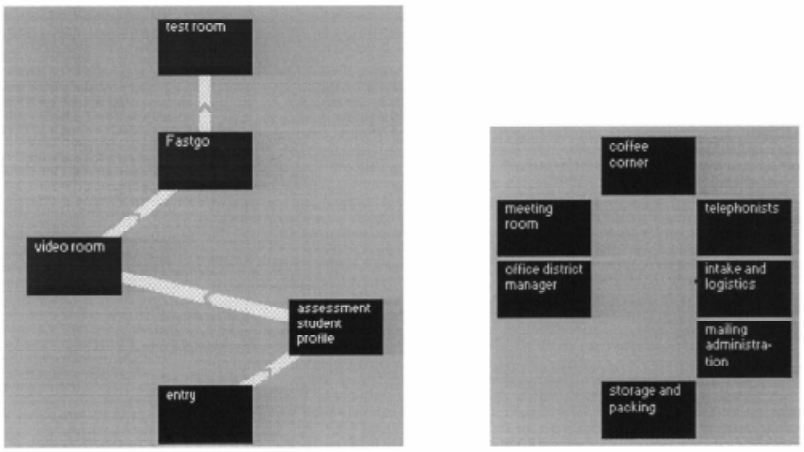

Fig. 3. A map of the program structure and of Fastgo

The Law Court program is a simulation of different law suits (Nadolski *et al.*, 1996) in which the student takes the role of a lawyer. The student has to apply the knowledge he has learned in the written materials of the law course in question to defend the defendant. Figure 4 gives the map and the arrangement of one of the rooms of the program.

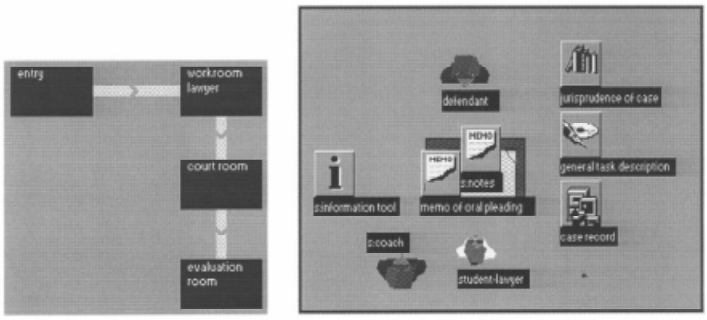

Fig. 4. Map and the arrangement of one of the rooms of the Law Court Simulation

Conclusion and future work

The approach dealt with in this paper for the specification of design has shown to be useful in practice. It provides an uniform way in which rather different designs can be specified. This makes designs comparable, accessible and easy to read. Furthermore, the re-use of objects is stimulated in the design approach

and this can lead to a more efficient production process and fewer bugs in programs.

It is important to remember here that the specification of a pedagogic design with *SHOC-TOOL* is only one step in the development and production process. Besides pedagogic design specifications, technical specifications have to be made, such as the design of: the structure of a program, the user-interface, the database, arithmetic kernels, content entry mechanisms and audio-visual components. When the design is complete it can be implemented using rules and house-styles for ensuring a *SHOC-TOOL* design is made uniformly and efficiently. The standard-object library plays an important role in this implementation process. After intensive tests the program can be reproduced, distributed and supported.

Some problems encountered by introducing the design specification approach and *SHOC-TOOL* are the following. It takes some time to understand the approach fully; appropriate training has to be set up. Also, experienced designers have already developed informal, *ad hoc* methods of their own. For them it is hard to integrate a new approach, especially when their own approach is more function-oriented instead of object-oriented. This is partly comparable to the problems of programmers switching from top-down structured approaches to more object-oriented approaches. Part of this problem is that the benefits of the approach are to be found in development teams rather than for individuals.

Some designers encounter the problem that the functional descriptions of a program are scattered around the design, attached to the different objects in the design. This is partly due to the technical implementation of *SHOC-TOOL*, but it is also inherent to the object-oriented way of working itself.

At the moment work is underway on the inclusion of some costing tools because the estimation of implementation costs is an important part of the design process. Furthermore the scope of the use of *SHOC-TOOL* is being broadened by using it for the design of complete Open University courses. Some work on this has been done already and it seems to be very promising.

References

Blair, G., Gallagher, J., Hutchison, D. & Shepherd, D. (eds.) (1991) *Object-Oriented Languages, Systems and Applications*. Pitman Publishing, London.

Booch, G. (1994) *Object-Oriented Analysis and Design with Applications.* The Benjamin / Cummings Publishers, Redwood.

Brown, J.S., Collins, A. & Duguid, P. (1989) Situated Cognition and the Culture of Learning. *Educational Researcher,* **18**, 1, 32-42.

Cates, W.M. (1994) *Designing Hypermedia Is Hell: Metaphor's Role in Instructional Design*. ERIC Document ED373706.

Choi, J. & Hannafin, M. (1995) Situated Cognition and Learning Environments: Roles, Structures, and Implications for Design, *Educational Technology Research and Development,* **43**, 2, 53-69.

Daal, M.M., Weges, H.G., Lelkens-Dreteler, C.E., Tonnaer G., Henzen, A., Nijboer, E. & van der Vegt, G.W. (1996) *Reflex.* Computer Program. Open university, Heerlen.

Dijkstra, E.W. & Fijstra, W.H.J. (1984) *Een methode van programmeren [a method of programming]*. Academic Service, Den Haag.

Dillenbourg, P. & Schneider, D. (1995) Mediating the mechanisms which make collaborative learning sometimes effective. *International Journal of Educational Telecommunications,* **1**, 131-146.

Duffy, T.M. & Jonassen, D.H. (1991) Constructivism: New implications for Instructional Technology? *Educational Technology,* **31**, 5, 7-12.

Hoogveld, A.W.M., Kampermann, A., van der Vlist, R., Steensma, H., Coors, P.M., van der Vegt, G.W. (1996) *IZEL.* Computer program. Open university, Heerlen.

Jackson, M. (1983) *System Development.* Prentice-Hall, New Jersey.

Jackson, R.B. (1995) Developing Formal Object-oriented Requirements Specifications: A Model, Tool and Technique. *Information Systems,* **20**, 4, 273-289.

Kearsley, G. (1995) Three Generations of Hypertext: lessons learned from the TIP Project. *Educational Technology Review,* **4**, 33-37.
(see also: http://gwis2.circ.gwu.edu/~Kearsley)

Kolb, D.A. (1984) *Experiential Learning.* Prentice Hall, New Jersey.

Koper, E.J.R. (1989) Een keuzestrategie voor de inzet van (electronische) media in het hoger onderwijs [a selection method for the use of (electronic) media in higher education]. *Tijdschrift voor Hoger Onderwijs,* **7**, 3, 78-88.

Koper, E.J.R. (1991) Inscript: a courseware specification language. *Computers & Education,* **16**, 2, 185-196.

Koper, E.J.R. (1992a) Premises for the Development of High Quality Educational Software. In *Book of Summaries of the European Conference on Educational Research* (eds. Tj. Plomp, J.M. Pieters & A. Feteris) pp. 634-637. University of Twente, Enschede.

Koper, E.J.R. (1992b) *Studieondersteuning met behulp van de computer [study support by means of a computer]*. Lemma, Utrecht.

Koper, E.J.R., Slootmaker, A. & Berkhout, J. (1993) *Shoc-tool.* Computer program. Open university, Heerlen.

Koper, E.J.R. (1995) PROFIL: a method for the development of multimedia courseware, *British Journal of Educational Technology,* **26**, 2, 94-108.

Lave, J. & Wenger, E. (1991) *Situated learning: legitimate peripheral participation.* Cambridge University Press, Cambridge.

Livergood, N.D. (1990-1991) Specification and Design Procedures, Functions, and Issues in Developing Intelligent Tutoring Systems. *Journal of Educational Technology Systems,* **19**, 3, 251-264.

Meyer, B. (1988) *Object-Oriented Software Construction.* Prentice Hall, New York.

Nadolski, R.J., Wöretshofer, J., Reijntjes, J., Kurvers, H. & Berkhout, J. (1996) *Law Court Simulation.* Computer Program. Open university, Heerlen.

Poncelet, G.M. & Proctor, L.F. (1993) Design and Development Factors in the Production of Hypermedia-Based Courseware. *Canadian Journal of Educational Communication,* **22**, 2, 91-111.

Scardamalia, M. & Bereiter, C. (1993/1994) Computer support for knowledge-building communities. *Journal of the Learning Sciences,* **3**, 3, 265-284.

Sommerville, I. (1985) *Software Engineering.* Addison-Wesly, Workingham.

Spector, J.M., Polson, M.C. & Muraida, D.J. (eds) (1993) *Automating Instructional Design: Concepts and Issues.* Educational Technology Publications, Englewood Cliffs, New Jersey.

Watabe, K., Hamalainen, M. & Whinston, A.B. (1995) An Internet based collaborative distance learning system: Codiless. *Computers & Education,* **24**, 3, 141-155.

Wirth, N. (1986) *Algorithms and Data Structures.* Prentice-Hall, New Jersey.