Specifying the Didactic Design of Educational Multimedia and Telematics applications

E.J.R. Koper
Centre for Educational Technology, Open university of the Netherlands,
Valkenburgerweg 167, 6419 AT Heerlen, The Netherlands
tel: ..31-45762657
fax: ..31-45762800
email: rob.koper@ouh.nl

Abstract

This article deals with the problem of the didactic (or 'pedagogical') design of educational software. An object-oriented design method and a supporting tool (called 'Shoctool') are presented. The method and the tool both stimulate the re-use of existing design and code segments. Resulting designs can be accessed for completeness, allow for comparisons among designs and are relatively easy to read for others than the designer himself. 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 of the didactic design of educational software in general. Shoctool is described along with several examples from real practice. The article concludes with a discussion of the experience we had with the introduction and use.

Published:

Koper, E.J.R. (1998). Specifying the Didactic Design of Educational Multimedia and Telematics Applications. *Journal of Computer Assisted Learning, 14,* 19-30.

1. Introduction

One of the problems in the development process of educational software, whether it concerns telematics applications, multimedia or classical CBI programs, is the issue of the didactic design of the program and the methods and tools to support this process. At the moment the available methods and tools does not balance the rich choices of instruments which are available at the technical level (including the numerous authoring systems available). In most projects the didactic design (also called 'pedagogical design') is done rather ad hoc. This results in design specifications which are rather difficult to compare with other designs and of which the completeness is difficult to access. The development process will be rather inefficient because of this. One of the problems of an ad hoc design approach is that the re-use of existing design elements and program code is not stimulated in the design process. This is especially a problem in an environment with a reasonable high volume of development and production of all different kinds of educational software.

At the moment most 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 there, the re-use of existing code will be restricted. One way to come out of this is to give the designers access to the existing object or class libraries. The problem however is that they tend to be rather technical and overwhelming for an instructional designer. The library should in fact be made visible in a view that designers can understand easily.

Another problem which makes the development process inefficient is that the technicians must make a translation of the rather loosely formulated functional specifications into formal object-oriented technical specifications. In this process the technical designer has to make a lot of interpretations and restructuring, which all have to be checked with the instructional designer. Often it occurs that the resulting program is not exactly what the instructional designer has meant. This may result in poor quality programs from the educational point of view.

To deal with this problem, we have set up a project at the Dutch Open university to design a method for the development of educational software which integrates instructional design methods and techniques and software engineering methods and techniques. The resulting method is described in several publications [1,2,3] and has been used for the design and production of several dozens of educational computer programs in the past years. In this article the focus is on one of the tools we developed, called 'Shoctool'[4]. With this tool an instructional designer can specify the didactic design of a program. First the idea of 'object-oriented didactic design' will be dealt with. Then a description of the program itself will be given in combination with examples.

2. Object-oriented Didactic Design

In the software engineering literature the concept 'object-oriented design' is not very strictly defined. Sommerville[5] distinguishes three types of design methods in this field: top-down structured design; data-driven design and object-oriented design. Top-down structured design is based on the work of Wirth[6] and Dijkstra[7], and is based on algorithmic decomposition to break down a complex system into subsystems. Data-driven design [8] is based on the idea of mapping inputs and outputs of a system. Object-oriented design [9,10,11] 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.

We have applied some of the concepts of object-oriented methods in software engineering to didactic design [12]. However, applied to this field some adaptations have to be made. For a more detailed description of the design approach we refer to Koper[1]. Only a short summary will be provided here.

The basic idea we had was the concept of an 'action-frame'. This is a model of an encapsulated time-space relationship, in which an actor interacts with certain objects in order to obtain certain learning outcomes. Situated in an educational context we refer to action-frames with the concept 'didactic scenario'. The object-oriented didactic design process is basically seen as a process of designing didactic scenario's.

The time-space relationship of the didactic scenario's 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 action frames - is to see the scenario's as 'virtual learning environments' or 'virtual rooms' in which students interact with objects. The same methaphor can be used for more time related scenario's. In that case the 'rooms' must be named to the central activity or the output generated by the activities, such as: 'preparation room' or 'design justification room'. The methaphor 'room' must not be taken to literally: also outside environments or imaginative environments are to be included.

We distinguished four different classes of objects in rooms: communication objects, tools, background objects and 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 can not be manipulated, but 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 this distinction in classes of objects we also make another distinction of objects, which has to do with the status of an object in the design. We distinguish: 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 the didactic 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 choice for the computer as the basic 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 of which is already decided in which medium 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 saves it; a programmer creates it when implementing the design. In 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 difference with standard objects is that the code of the objects isn't optimised for re-use and adaptations and that it will probably take more effort to adapt the code to specific needs. When a certain object is re-used more than once, one can decide to make a standard object of it.
Apart from a distinction in classes and in the design status of objects we can also make a distinction in the view of the objects. We distinguish four possible views on objects (depending on the kind of object): a didactic view, a content related view, a display view and a technical view. A good design and development environment must make it possible to allow for al these views on the same objects, so that an instructional designer is able to view the object from the didactical point of view, a content matter specialist can look at the content of objects, a graphical designer (or audio-visual producers) 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.
In the next paragraph the use of these concepts in the didactic design process with the help of Shoctool will be further explained.

3. Shoctool: a description

Shoctool is a tool to support the instructional designer to specify the didactic design of an educational application. This support is three-fold: 1) it gives information about the design process and the steps to be taken; 2) it contains editors and tools for the different components of the design; 3) it has libraries of standard objects.
The tool supports the designer in his task, but does not think for the designer: it is not an expert system nor a decision support system (in the quantitative sense).
The didactic design process comprises of 5 steps all supported by Shoctool:

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.
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, e.g.)? What delivery medium will be used: CD-ROM, diskettes?
3. A choice of an appropriate didactic model must be made. A didactic model 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 e.g. collaborative learning [13,14] or experiential learning [15].
4. The design of the didactic scenario.
5. The selection of the media in which the functional objects will be implemented and the selection of communication codes for all functional objects.

For management purposes a 6th step is included in Shoctool: the justification of the design.

Shoctool 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. The tool has seven rooms (entry, data, model, design, implementation, justification, print) each of which will be explained now.

The entry room

After starting the program the designer will come in the 'entry room' of the program where he meets a porter who explains the objectives of Shoctool 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 step 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 a selection of a didactic model (step 3) is to be made. The question which must be answered by the designer is: 'Which didactic 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 his own didactic model, he must sample the above information as much as possible himself.

The design room

The next room is the 'design room', which is the most important room in the program. In this room the designer is asked to make a design of the didactic scenario (step 4), which is directed at the learning objectives and target group characteristics and which is

inspired by the principles of the didactic model. In essence this is a creative process: Shoctool gives the possibility to structure this 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 fulfilment of the total set of activities in the different rooms should lead 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 [16] (in technical realisation at the moment) is given in figure 1.

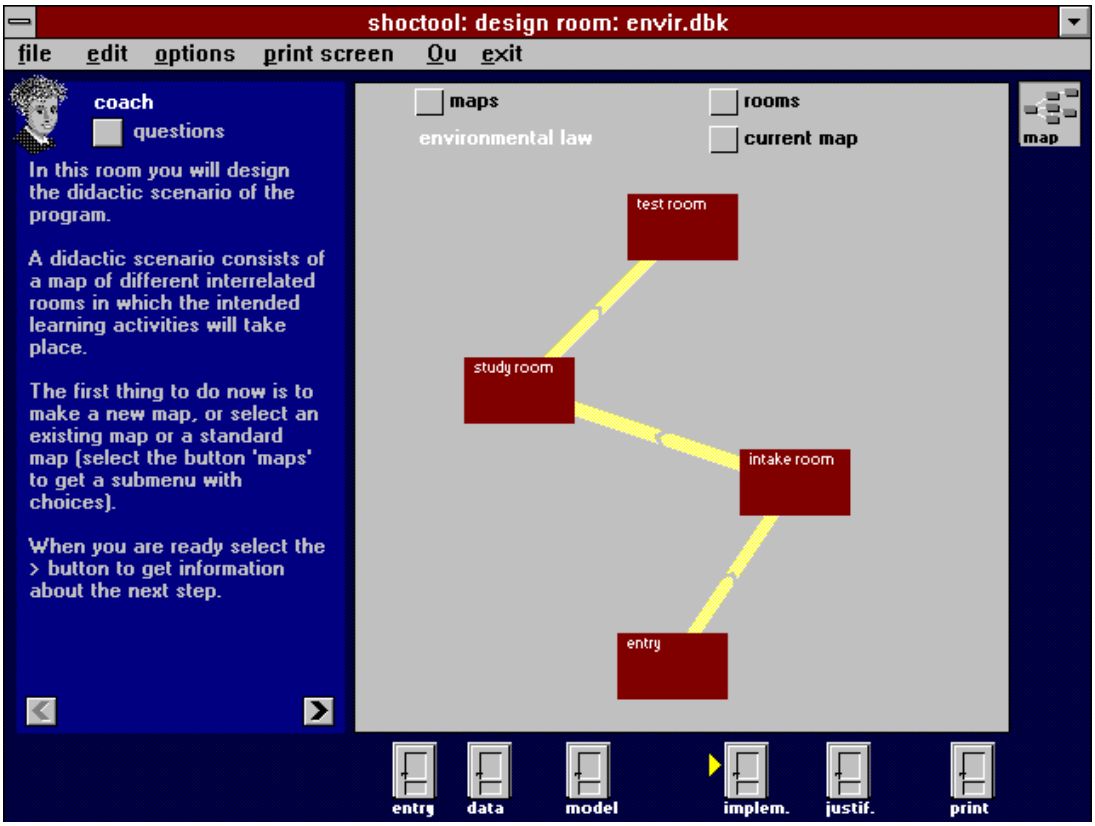

Figure 1. An example of a map designed with Shoctool

The map of the didactic scenario is in the field to the right. It 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' he will get a text editor in which the designer is asked to specify the function of the room.

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 students background (most are adult students) and study needs are accessed 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' in the menu 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.

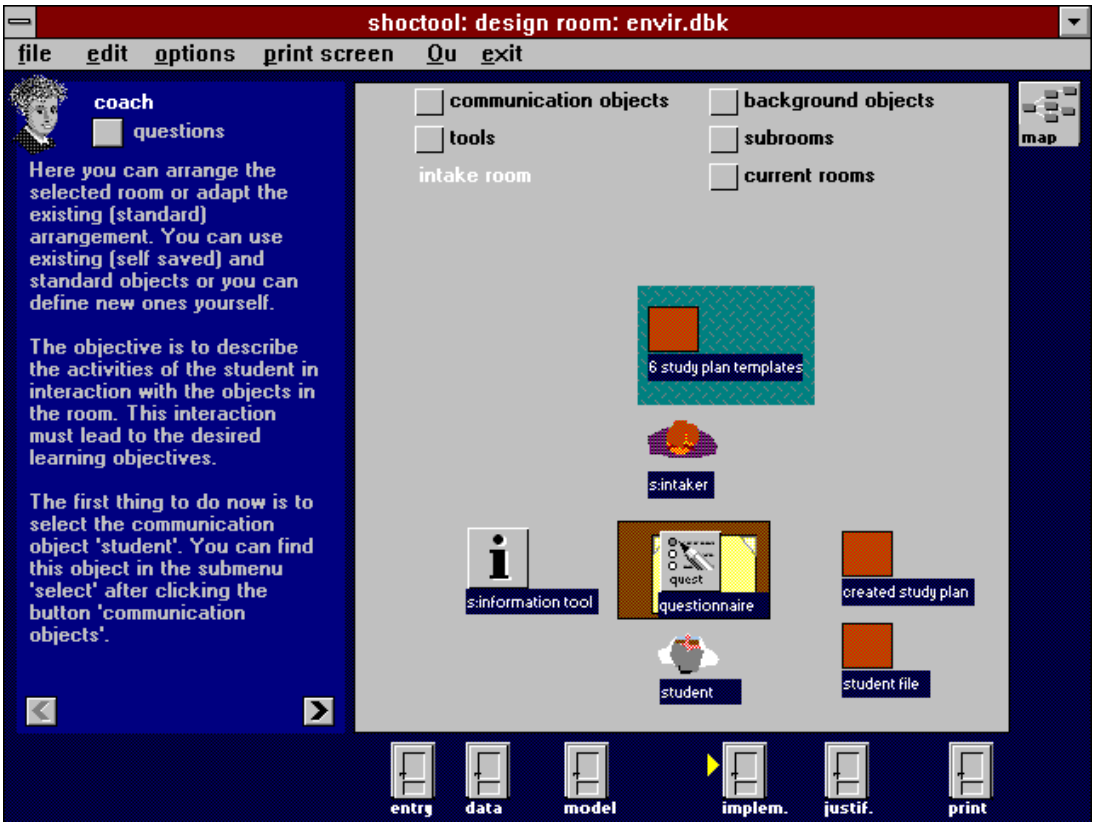

Figure 2. Arrangement of the intake room

When the intake room was entered for the first time it was 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 are needed to interact with. The editor in the room allows to add three types of objects - 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 arranges 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 has 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 Ou programs.
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 described, the implementation of the different objects has to be decided for. 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. Furthermore the designer can directly go 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 for instance 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 in the education. In this room the designer can also reconsider the implementation of objects and make changes where appropriate.

The justification room
The justification room provides an text editor in which the designer can type his arguments 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[17].

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.

4. Example: Exercises in Micro and Macro Economics

The example given above about environmental law is just one of the possible designs to be specified with Shoctool. In this and the next paragraph some more examples will be given. First a very straight forward design will be presented: the design of a program which provides exercises in the field of micro and macro economics [18]. The program has three rooms: an entry room, an exercise room and a print room (see figure 3). The entry room has the same function as in the previous (and the next) example and consists of a lot of standard elements. The exercise room consists of a tutor whose function is to guide the student, to present the exercises and to provide feedback. Furthermore a 'graph tool' is present in the room: this is an adapted standard object to graph functions which can be entered as an expression. Students must use this tool to solve most of the exercises. In the print room the student can select different kinds of prints.

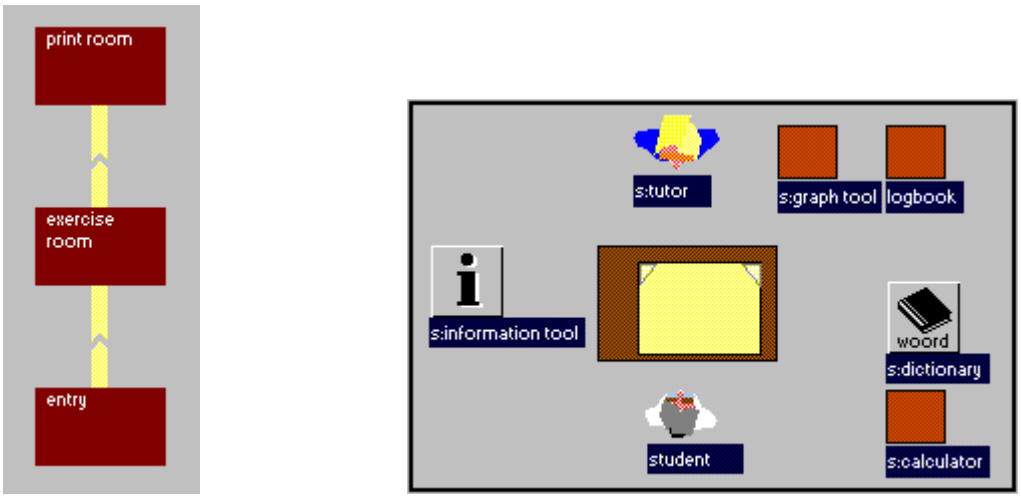

Figure 3. Map and Arrangement of the exercise room of the Micro and Macro Economics program Examples: simulations of authentic environments

Didactic models which attract much attention recently are the constructivistic approaches [19,20,21]. The idea 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[22] would call it. Another possibility is to let students interact with each other in the environment according to collaborative learning ideas.
Shoctool 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 [23] - 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: a) by filling in a test about leadership and b) 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 packing 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 on his own, learned in a simulated authentic environment. The Shoctool design comprises of two maps: one of the program structure and one of Fastgo (figure 4)

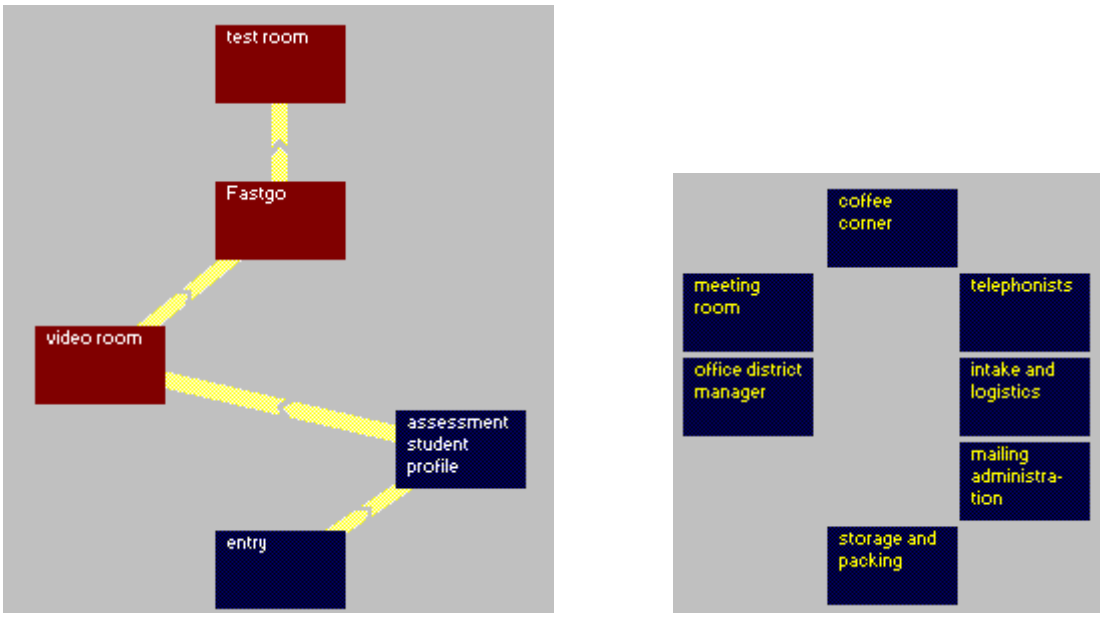

Figure 4 map of the overall program structure (left) and the submap of Fastgo

To get an impression of one of the rooms, figure 5 shows the office of the district manager (the role played by the student). The program is implemented as a series of production rules, each describing what should be done in every possible situation. The rooms and actions are implemented with MPEG full screen video (including audio).

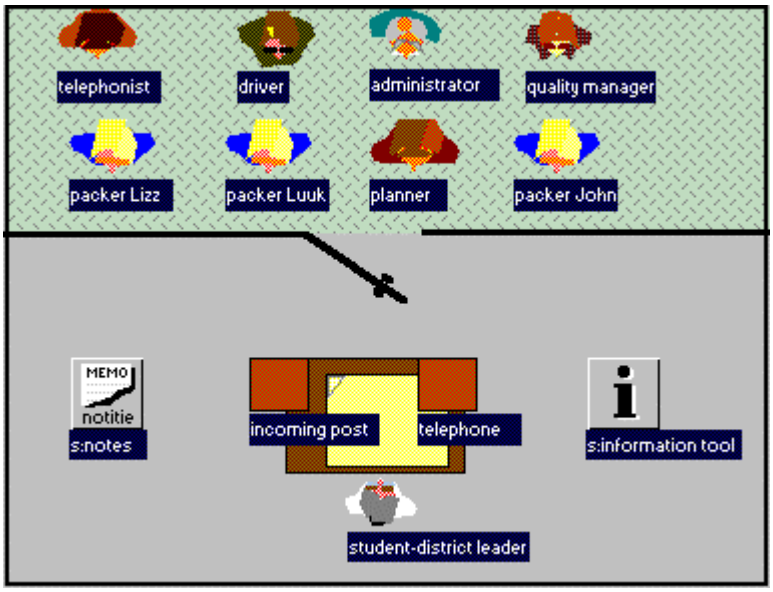

Figure 5. Arrangement of the district manager's office

Comparable examples which are designed with Shoctool are a program on Marketing Management [24] and a program which simulates a Law Court [25]. In the Marketing Management program (a multimedia CD-rom) the student is in the role of an assistant marketing manager, who assists the virtual marketing manager in setting up marketing strategies for a touristic firm. The virtual marketing manager acts as a coach and expert. The program consists, among other things, of a simulation in real time and a forecasting tool: the management information system. One of the design considerations was to implement the work environment of a marketing manager as authentic as possible. The simulation is for instance based on the results of a real firm. Figure 6 gives the map of the program.

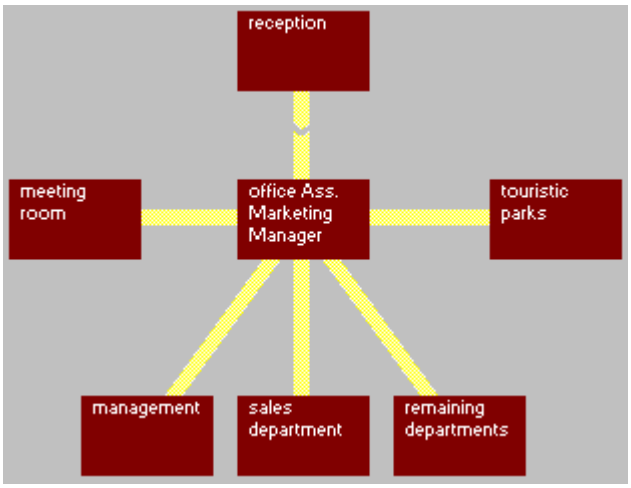

Figure 6. Map of the Marketing Management program

The Law Court program is a simulation of different law suits 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 7 gives the map and the arrangement of one of the rooms of the program.

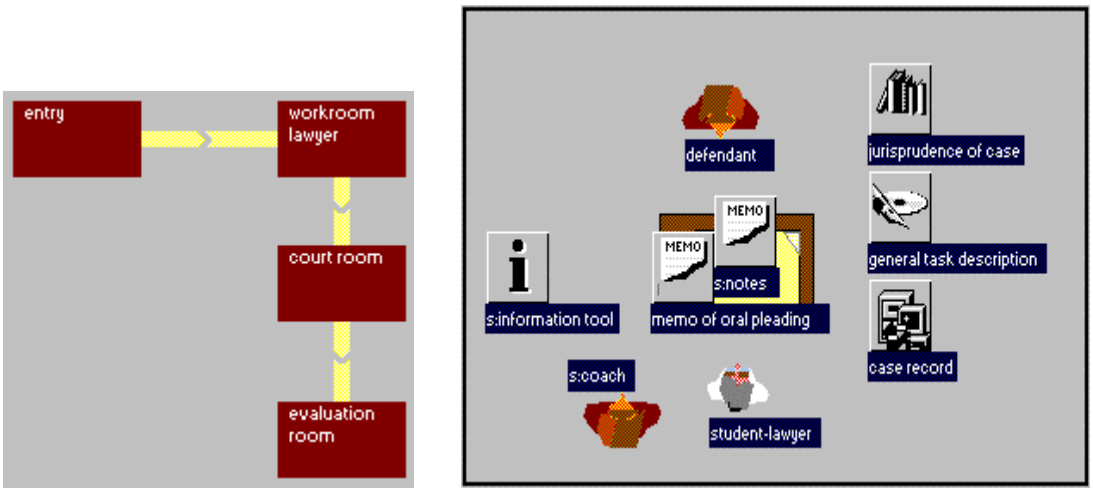

Figure 7. Map and the arrangement of the workroom of the Law Court Simulation

In the above mentioned examples some of the same (standard) objects are used, e.g. the objects of the entry room, the notes object, the information tool and the coach.

5. Conclusion and future work

The object-oriented approach of didactic design has shown to be useful in practice. This is especially true for the uniform way in which rather different designs can be specified, which makes designs comparable, accessible and easy to read. Furthermore, the re-use of objects is stimulated in the design approach which can lead to a more efficient production process and less bugs in programs.

It is important to remember here that the specification of a didactic design with Shoctool is only one step in the development and production process. Besides a didactic design - for instance - also a more technical oriented design has to be made, such as: the design of the technical structure of the program, the user-interface design, the database design, the design of arithmetic kernels, the design of content entry mechanisms and the design of audio-visual components. When the design is complete it can be technically realised. In the Ou we have developed rules and house-styles for implementing a Shoctool design uniformly and efficiently. The standard-object library plays an important role in this implementation process. After intensive tests the program can be reproduced, distributed, supported and so on. See Koper[1] for an overview of the whole process.

Some problems we have encountered by introducing the object-oriented design approach and Shoctool are the following. It takes some time to understand the approach fully, appropriate training has to be set up. Also, most experienced designers have already developed a design practice of their own. For them it is somewhat hard to integrate a new approach, especially when their own approach is more function-oriented instead of object-oriented. This is 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 presented here are to be found especially on the group level and to a lesser extend on the individual level. Sometimes designers have for instance problems with an unchangeable part of a standard object. However, using it the way it is will make the overall process more efficient.

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. Partly this is due to the technical implementation of Shoctool - it is for instance possible to make an

integrated view on all the functions of all the objects - but partly it is inherent to the object-oriented way of working itself.
At the moment we are working on the addition of some costing tools to Shoctool, because the estimation of realisation costs is an important part of the design process. Furthermore we are at the moment broadening the scope of the use of Shoctool by using it for the design of complete Ou-courses. Some work in this field has been done already and seems to be very promising.

References

1. Koper, E.J.R., PROFIL: a method for the development of multimedia courseware, *British Journal of Educational Technology* ***26*****,** 2, 94-108 (1995).
2. Koper, E.J.R., Premises for the Development of High Quality Educational Software, In Tj. Plomp, J.M. Pieters & A. Feteris, Book of Summaries of the European Conference on Educational Research (pp. 634-637), University of Twente, Enschede (1992).
3. Koper, E.J.R., Inscript: a courseware specification language, *Computers & Education* ***16****,* 2, 185-196 (1991).
4. Computer program designed by: E.J.R. Koper, tech.realisation: A. Slootmaker, graphical design: J. Berkhout. Program is in Dutch, screendumps are translated for this article.
5. Sommerville, I., *Software Engineering,* second edition (p. 68), Addison-Wesly, Workingham, England (1985).
6. Wirth, N., *Algorithms and Data Structures.* Prentice-Hall, Englewood Cliffs, NJ (1986).
7. Dijkstra, E.W. & Fijstra,W.H.J., *Een methode van programmeren [a method of programming]*, Academic Service, Den Haag (1984).
8. Jackson, M. *System Development.* Prentice-Hall, Englewood Cliffs, NJ (1983).
9. Booch, G., *Object-Oriented Analysis and Design with Applications,* second edition, The Benjamin/Cummings Publ. Comp., Redwood, California (1994).
10. Meyer, B., *Object-Oriented Software Construction*. Prentice Hall, New York (1988).
11. Blair, G., Gallagher, J., Hutchison, D. & Shepherd, D. (Eds.), *Object-Oriented Languages, Systems and Applications,* Pitman Publishing, London (1991).
12. Koper, E.J.R. *Studieondersteuning met behulp van de computer [study support by means of a computer]*. Lemma, Utrecht (1992).
13. Watabe, K., Hamalainen, M, Whinston, A.B., An Internet Based Collaborative Distance Learning System: Codiless, *Computers & Education* ***24***, 3, 141-155 (1995).

14. Scardamalia, M. & Bereiter, C., Computer support for knowledge-building communities, *Journal of the Learning Sciences* ***3****, 3,* 265-284 (1993/1994).
15. Kolb, D.A., *Experiential Learning,* Prentice Hall, New Jersey (1984).
16. Didactic design and project management by M.M. Daal, H.G. Weges & C.E. Lelkens-Dreteler, Tech. realisation: E. Nijboer & G.W. van der Vegt, content matter specialists: F.P.C.L. Tonnaer & A. Henzen.
17. Koper, E.J.R., 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 (1989).
18. Didactic design and project management by P.P.M. Van Vilsteren, Tech. realisation: H. Kurvers, content matter specialist: G.M. Popelier, graphical design: J. Berkhout.
19. Brown, J.S., Collins, A. & Duguid, P., Situated Cognition and the Culture of Learning. *Educational Researcher* ***18,***1, 32-42 (1989).
20. Duffy, T.M., & Jonassen, D.H., Constructivism: New implications for Instrucional Technology? *Educational Technology* ***31***, 5, 7-12 (1991).
21. Choi, J., Hannafin, M., Situated Cognition and Learning Environments: Roles, Structures, and Implications for Design, *Educational Technology Research and Development* ***43,*** 2, 53-69 (1995).
22. Lave, J. & Wenger, E., *Situated learning: legitimate peripheral participation*. Cambridge university press, Cambridge (1991).
23. Didactic design and project management by A.W.M. Hoogveld, tech. realisation: P.M. Coors & G.W. van der Vegt, subject-matter specialists: A. Kampermann, R. van der Vlist, H. Steensma, audio-visual production: Available.
24. Didactic design and project management by R.J. Nadolski, tech. realisation: A.Slootmaker & M. Jordense, subject-matter specialist: P.J. Huysse; graphical design: J. Berkhout, audio-visual design: W. Kerstjens, audio-visual production: Available.
25. Didactic design and project management by R.J. Nadolski, tech. realisation: H. Kurvers, subject-matter specialists: J. Wöretshofer, J.M. Reijntjes, graphical design: J. Berkhout.