An Object-Oriented Approach to the Rapid Prototyping of Courseware: Instructional Design Considerations and Method

by

Toh Seong Chong
Centre for Educational Technology and Media,
Universiti Sains Malaysia
11800 Penang, Malaysia

Tel: 604-6577888 ext 3431; Fax: 604-6576749;

E-mail: tohsc@usm.my

Paper presented at :
6th World Conference on Computers in Education
(WCCE 95)
Birmingham, United Kingdom
(23 - 28 July, 1995)

* This project is funded by grants from the Chancellery, Universiti Sains Malaysia

Abstract

The object-oriented programming paradigm (OOP) has revolutionized the software development process. This paper dicusses the instructional design considerations in the development of a shell using the OOP paradigm to enable courseware authors to create courseware rapidly through rapid prototyping. This shell has been successfully developed at the Centre for Educational Technology and Media, Universiti Sains Malaysia. It has the flexibility of incorporating a variety of computer-based learning modes ranging from tutorials which are essentially "frame-oriented" representation style, to simulations which basically use "model-oriented" representation style. This paper further describes the instructional design considerations in developing this shell and the various courseware that have been developed by teachers using this method.

Theme:	**Software; Information Technology**
Level:	**Secondary Education; Higher Education**
Topic:	**Design; Educational Technology**
Secondary topics:	**Authoring Systems; Computer-Assisted Instructions**

Introduction

In the last ten years, advances in the microelectronics technologies have successfully achieved more computing power and data storage at increasingly low cost. These advances have led to a rapid proliferation of microcomputers amongst all sectors in the society, including the use of microcomputers for computer-based learning. This has led to a scenario where is an urgent need for trained instructional developers to develop quality courseware. Unfortunately trained instruction developers are in short supply.

Issues and Challenges

Instructional Systems Development (ISD), especially for computer-based courseware, is too labour intensive usually requiring more than 200 hours of development for a single hour of instruction. Moreover the traditional ISD model is not adequate for computer-based interactive courseware instructional development because it provides little guidance for interaction and it does not specify an adequate syntax for knowledge representation [1].

Teachers find it hard pressed for time to learn a programming language or software engineering principles because they find it almost impossible to squeeze precious time from their already hectic teaching schedule. In order to encourage teachers to develop courseware on their own, there is a need to reduce the development of courseware to delivery ratio by at least an order of magnitude -- from 200:1 to 20:1. In order to meet this challenge, there is a need to provide tools which empower subject matter experts to do effective computer-based instructional development without requiring them to have extensive training in instructional design or authoring systems.

The Instructional Systems Development (ISD) Approach in courseware production

The instructional systems development (ISD) is the most widely used and oldest paradigm for courseware production. It starts with the planning phase, needs analysis, design, development and implementation [2]. Each of the earlier phases produces intermediate outcomes that are used in the succeeding phases. For example, analysis produces job/task descriptions, design produces an objectives hierarchy, an early stage of development produces a learning activities description, and later stages produce storyboards, scripts and computer code.

The ISD approach is based on an assumption that no concrete product will be available until late in the process, yet there is a need to manage and control the development to ensure that the process stays on track; hence the well-defined phases, steps and succession of intermediate outcomes characteristic of ISD. Two additional assumptions are implicit in this approach. First is the assumption that an adequate evaluation of the progress of the design is possible using the abstract, intermediate products of the early phases. Second is the assumption that the phased of ISD are relatively independent, with outputs of one phase being inputs to the next, allowing development to be stabilized at break points between phases.

However, over the years, criticism of the paradigm has caused even active supporters to question its applicability in all situations [3].. This is because the paradigm deals with problems in a sequential and linear fashion which real life software projects rarely follow [4].Another problem of the ISD approach is that the software designer (usually the teacher) is actually introduced to the product only after the implementation so that change requests are likely to build up from that point. See Fig. 1. Thus more often than not, a significant gap exists between the software (or the teacher's) expectations and the product capabilities.

<insert Fig. 1 here >

Development of a flexible authoring System

What is a possible solution? We suggest the development of a flexible authoring system that incorporates an instructional transaction shell with the following important features:

* *Caters for different levels of authoring expertise* -

The authoring system should be adaptable to authors having different levels of authoring expertise. Three levels of authoring could be assumed: low, middle and high level. Low-level authors will be able to develop courseware by selecting one of the predefined instructional templates and carrying out prompt-driven or menu-driven conversation with the system. Middle-level authors will be enabled to adapt instructional strategies by using the build-in authoring language in the system. The high-level authors will be facilitated to reorient the application of the system by creating or modifying the domain-related knowledge-base.

* *Domain knowledge representation* -

The authoring system should be flexible enough to cater for various domains of knowledge representation formalisms ranging from the traditional frame-oriented representations formalisms to model-oriented simulations. This transaction shell approach [5] underlain by the second generation instructional design theory [6] standsfor an effort towards this goal.

* *Capable of rapid prototyping of courseware* -

The authoring system should provide tools to enable courseware developers to have sufficient functionality and usability to get the developers started. The prototype should

"work" from the start in order to deliver value to the users as soon as possible. Hence providing users with a prototype which works right from the start, helps focus their attention on task-related issues first and overall appearance of the system second. By quickly providing the developers with a basic shell to start with, this will enable them to apply their experiences to iteratively evolve the system.

Rapid Prototyping

Prototyping is a process of creating a model of the software by the developer. This prototype is just an executable version of a product which has the key elements of the final version but which is incomplete in many respects, for example, in terms of functionality and robustness. With a prototype, software designers and developers can actually see what is possible and how their requirements translate into software.

Rapid prototyping [7, 8] is a process of quickly building and evaluating a series of prototypes. This method of courseware development requires the availability of tools that offer modularity [9]. It allows one to create and test input designs, output designs, and simple procedures. See Fig. 2.

< insert Fig. 2 here >

The advantage of rapid prototyping is that the designer may experiment with and evaluate a number of design approaches before committing to one for further development. In order for rapid prototyping to be practically useful, it may be possible to generate the prototypes with minimal investment. Thus the primary advantage of prototyping is that it provides the designer with concrete feedback in terms of final product, as compared to the more abstract feedback provided by the conventional products of analysis and design.

The ID_2 Rapid Prototyping Development Model

An instructional design model proposed by Merrill [6], also known as second-generation instructional model (ID_2) serves as underpinnings for rapid prototyping. The model comprises of seven steps (see Fig. 3). Knowledge analysis is the acquisition and representation of the subject matter content using a knowledge representation model [11]. Audience and environment analysis identifies general characteristics of the learners and the instructional setting. Strategy analysis selects and sequences transactions to instruct the content. Transaction configuration involves setting parameters for transactions to customize their behaviour. These four steps are highly interactive in that they all use a single representation of the content, and each step responds to and creates requirements for the other steps.

<insert Fig. 3 here>

Transaction detailing is the generation of graphics, animation, voice and text screens as required by the transaction shells and the content. The implementation and evaluation phase is the actual delivery and assessment of the courseware.

Thus the step in ID_2 development model roughly parallel those in ISD. The key differences are the products and the interaction between the steps. The products developed in the early phases are precisely those products that will be carried all the way through the development cycle, including the delivery of instruction. This is in sharp contrast to the progression of abstract intermediate products, translated one into another, that is seen in ISD. These products, and the concrete feedback they provide early in the development process as a result of the rapid prototyping approach they support, are the key differences between ID_2 and ISD.

The CETM, USM Research Project

At the Centre for Educational Technology and Media, USM we attempted to create a transaction shell that allows development of courseware through rapid prototypng. We called this transaction shell the Toh/Abdul Rahim (TAR) Authoring [12]. This transaction shell is created in DOS environment using "object-oriented" programming of C++. Within the shell is a knowledge base which may be in the form of text, graphics, sound and animations. "Objects" in this knowledge based is linked together immediately from one part of the information to another. This user-friendly environment allows quick prototyping of classroom software by educators who do not possess knowledge of computer language such as Pascal, BASIC or C++.

The Architecture of The Toh/Abdul Rahim (TAR) Authoring

The Toh/Abdul Rahim (TAR) Authoring consists of 4 components, namely the text editor, the graphics editor, the quiz generator and the courseware organizer. It is a flexible authoring environment comprising of three levels of interface namely the author interface, the instructor interface and the user interface. The author interface refers to the environment which enables the access to the source code to customize special needs of the instructional designer or instructor. For example modifications can be made to the source codes for special simulations, animations etc. The instructor interface refers to the platform where the instructor can make use of the existing shell to design a courseware based on the storyboard created. Three main components are automatically generated by the TAR Authoring, namely menu design, buttons and text layout. The user interface refers to the final product of the courseware where the user

could use it to learn a particular topic. This consists of the lesson presentation, formative evaluation and sumative evaluation. See Fig. 4

<insert Fig. 4 here>

The Text Editor

The text editor in the TAR Authoring enables the instructor to open a new text file in the shell, or edit an existing text file. Up to 10 files can be opened at any one time. The courseware designer could move from one file to another by clicking the appropriate window-text. Another feature of the text editor is that the instructor can highlight the key words in the text by putting attributes to the word with the symbol symbols { }. The text begins with Topic 1 follow by Topic 2 and so on. Subsume under Topic 1 will be sub-topics given the names Topic 11, Topic 12 and so on. The same applies to Topic 2, where subsume under it will be sub-topic Topic 21, Topic 22 and so on. See Fig. 5

<insert Fig. 5 here>

The Graphic Viewer

Graphics can be created using DOS-based Graphics programme such as Harvard Graphics, PC PaintBrush or Corel Draw and subsequently screen-captured as PCX files by using any transient-stay-resident (TSR) graphics programma such as GRAB in WordPerfect or CAPTURE in Harvard Graphics. The Graphic Viewer will enable the author to view the graphics PCX files captured.

The Course Organizer

This program automatically manages the main menu display, the sub-menu display, the placement of graphics and text files. It also organize the formative evaluation questions. By

arranging the text and graphic files in the sequence required by the instructor, the course organizer will automatically generate the courseware. Answers to the formative evaluation specified here is automatically incorporated into the courseware.

The Courseware generated by the TAR Authoring

The courseware generated by the TAR Authoring has a motivation opening screen to arouse learner's interest (Fig. 6); a main menu (Fig. 7), and sub-menu with navigation buttons (Fig.8). Simulations are inserted in the courseware as and when required (Fig. 9). It has also formative evaluation question (Fig.10) and dynamic database comparison (Fig. 11)

< insert Fig. 6 to Fig. 11 here >

Using this authoring tool created, over 40 different courware have been successfully created at the CETM, USM with topics ranging from English Language, Science , Mathematics and Living Skills. Every couseware incorporates instructional strategies necessary to enhance learning.

Conclusion

This paper identifies some of the issues and challenges in the development of quality courseware. It also discuss the limitations of the ISD model in designing courseware It also discuss how instructional design principles of the ID_2 model which can be used for rapid prototyping of courseware. Development of an authoring tool that allows rapid prototyping represents a new generation of authoring tool, which is expected to have a higher degree of flexibility than traditional authoring tools. The TAR Authoring tool developed in the CETM, USM stands for a pilot effort towards this flexible authoring tool. It has the flexibility of incorporating instructional strategies pertaining to the tutorial line which is a "frame-based"

representation style as well as other instructional strategies pertaining to the simulation line which basically use a "model-based" representation style.

We do not yet dare to conclude that flexible authoring tools will be the major stream of authoring environments in the near future. Nevertheless, we believe that further research into flexible authoring tools could contribute significantly to the development of more powerful and useful environments to produce quality courseware and to a wider of computer-based learning.

Reference

[1] Merrill, M. D, and ID_2 Research Group, (1994). *Automated Instructional Design and Development* , Paper presented at the Asia Pacific Information Technology in Training and Education Conference and Exhibition, Brisbane, Australia. 28 June - 2 July,1994

[2] Branson, R. (1975). *Interservice procedures for instructional systems development.* Tallahassee, Fl: Florida State University.

[3] Wong, Simon. C.H. (1993), Quick prototyping of educational software: an object - Oriented Approach. *Journal of Educational Technology Systems*, 22:2, pp. 155-172.

[4] Silver, G.A. and Silver M.L, (1989). Systems analysis and design, Addison - Wesley, Reading, Massachusetts.

[5] Zonghmin, L and Merill, M. D (1990) Transaction shells: a new approach to courseware authoring. *Journal of Research on Computing in Education, 23, 1, 72 - 86.*

[6] Merrill, M. D., Zonghmin, L. and Jones, J. K. (1990) The second generation instructional design research program. *Educational Technology,* 30, 26 - 31.

[7] Institute of Electronic and Electronic Engineers (IEEE), (1989). Rapid prototyping of software development, *Computer,* 22:5 [special Issue]

[8] Luqi (1988). Knowledge-based support for rapid software prototyping. *IEEE Expert,* 3 (4), 9-19

[9] Tripp, S.D. and Bichelmeyer, B., (1990). Rapid prototyping: An alternation instructional design strategy. *Educational Technology Research and Development*, 38:1, p. 31-44

[10] Toh, S. C., & Abdul Rahim M. S., (1994). *The Design and Construction of an Authoring Tool to enhance Learning.* Paper presented at EDUCOMP '94, Universiti Sains Malaysia, Penang. 14 - 16 June, 1994

[11] Jones, M. K. Zonghmin, L. and Merill, M. D. (1990) Domain knowledge representation for instructional analysis. *Educational Technology,* 30, 10, 7 - 32.

[12] Schwen, T. M., Goodrun, D. A., & Dorsey L. T. (1993). On the design of an enriched learning and information environment (ELIE). *Educational Technology*,1993, 33 (11), p. 5 - 20

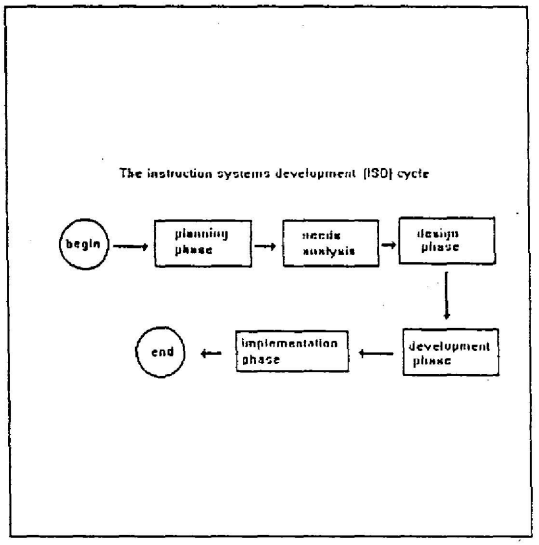

Fig 1. The ISD cycle

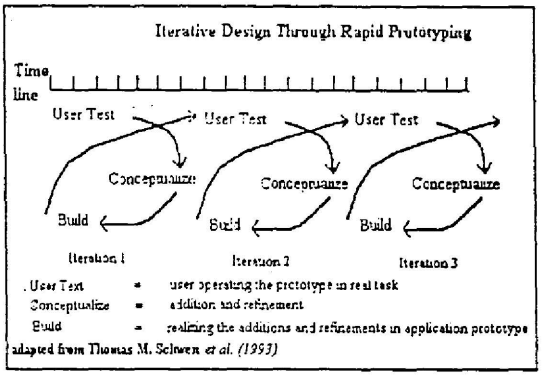

Fig. 2. Rapid Prototyping with iteration

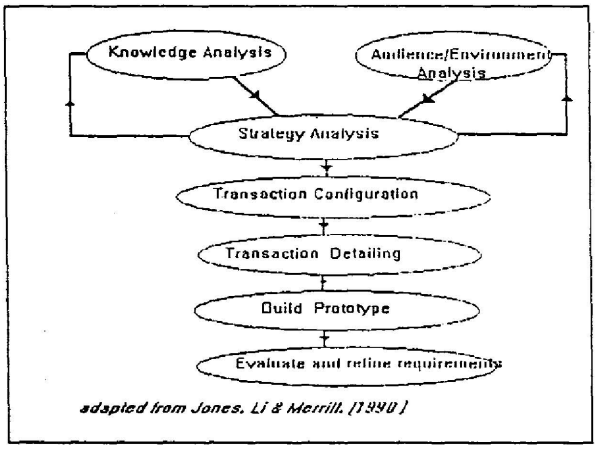

Fig. 3. ID_2 Model for Rapid Prototyping

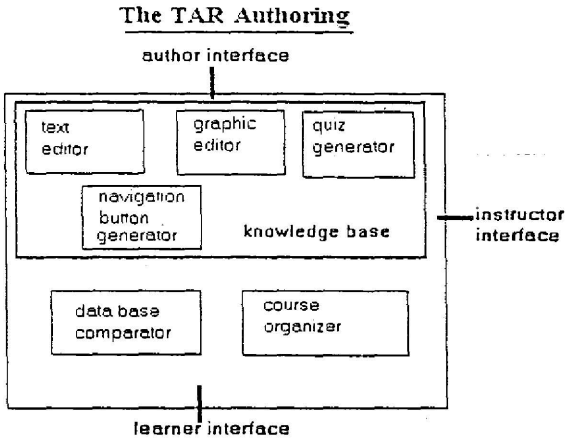

Fig. 4 Different Interfaces in the TAR Authoring

Organization of text files in the courseware produced by tsc/arms authoring

TOPIC1	TOPIC2		TOPIC7
subtopics			
TOPIC11	TOPIC21	....	TOPIC71
TOPIC12	TOPIC22	....	TOPIC72
TOPIC13	TOPIC23	....	TOPIC73
TOPIC14	TOPIC24	....	TOPIC74
TOPIC15	TOPIC25	....	TOPIC75
..	....		.....
..	....		.....
..	....		.....

Fig.5. The structure of the text in the courseware

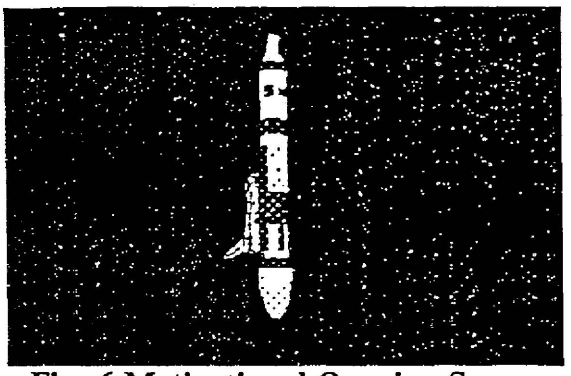

Fig. 6 Motivational Opening Screen

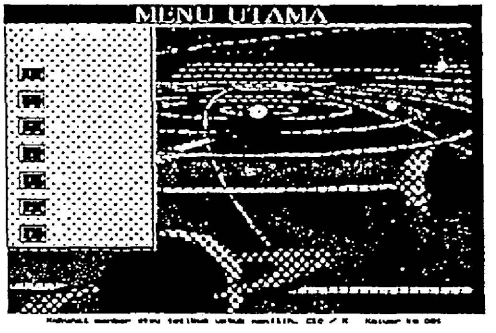

Fig. 7. Main Menu

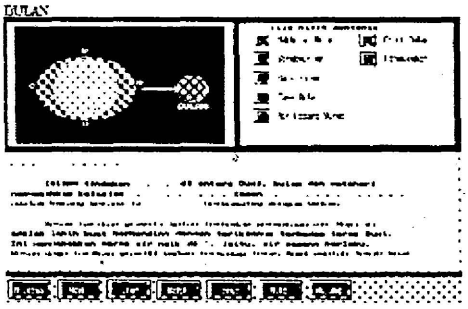

Fig. 8. Sub-Menu with navigation buttons

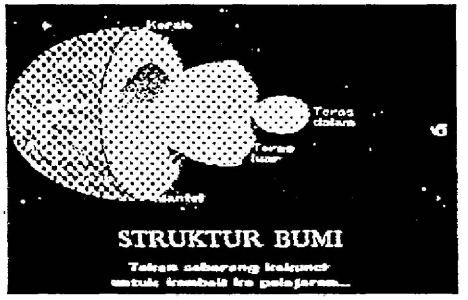

Fig. 9. Model-based simulation

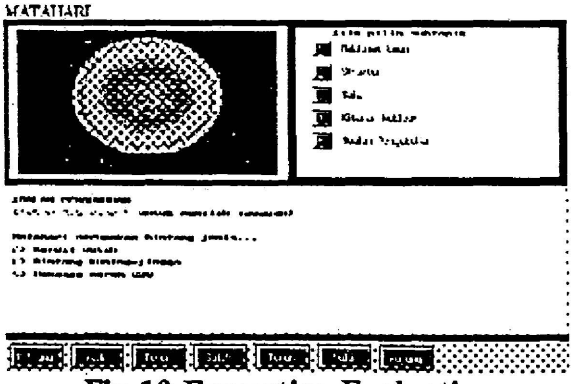

Fig.10 Formative Evaluation

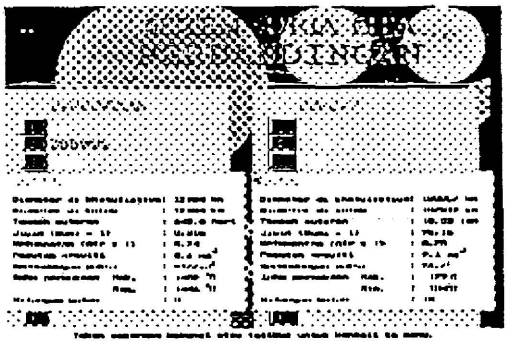

Fig. 11. Dynamic database