COURSEWARE DESIGN SUPPORT

Robin Johnson

Dept of Mathematics and Computer Science
University of Technology, PMB,
Lae
Papua New Guinea

Introduction

Courseware design, like all design problems, is a complex task. It presents the same kinds of problems that other design problems present. It is under constrained, it has many possible solutions none of which is optimal, it is not easily evaluated at design- time, and the requirements do not suggest an obvious structure for the solution (Guindon 1990; Goel & Pirolli 1992). Additionally, courseware designers are expected to possess an extensive model of how users will learn from courseware, and hence what makes effective courseware. Even with traditional frame-based, linear courseware there was a need to provide support for courseware designers but with the advent of knowledge-based courseware design tools the need for support is even greater.

Knowledge-based courseware design tools, such as, DISCourse (DISCourse 1994), and KAFITS (1991), utilise multiple knowledge sources to represent content knowledge, instructional knowledge, and student modelling knowledge. They provide some kind of adaptation to the learner's performance, and the content that is being taught and they allow the designer to implement a wide range of tutoring styles, many more than traditional courseware allows. They also provide sophisticated forms of interaction between the learner and the software, including the use of multimedia. These tools do not produce Intelligent Tutoring Systems (ITSs) but they do provide courseware which is much more flexible, reusable, and adaptive than traditional courseware.

This paper describes the investigation of a support tool for the DISCourse project which uses case-based advice. It also posits some general ideas about the use of case- libraries to support software design activities.

A Courseware design problem

The main problem that was investigated in this research was how to support designers in the production of courseware that adapted to learner attributes, such as, prior knowledge, preferred level of control, motivation, learning style etc. I don't wish to argue the merits of this kind of adaptation, except to note that it is much simpler to implement than cognitive adaptations of the kind used by ITSs. There has been a lot of research into adaptations to learner attributes, both in the form of empirical psychological research, and less formal investigations of adaptive courseware. The results from this research are incomplete, and contradictory. However, some of the findings from this work have become well established as principles which guide courseware development, such as the findings which relate locus of control, prior knowledge and levels of ability. Other research has been used less widely, such the findings about adaptations to learning styles, and motivation.

DISCourse decided to adapt to multiple learner attributes, an approach which has not been tried before because traditional courseware has not been powerful enough to implement it. Adaptation to multiple attributes introduces several problems of complexity. Firstly, there will be too many combinations of attributes in the learner population to adapt to the learners individually and there are no established principles about how to group learners. Secondly, there is no established knowledge about how to combine adaptations, and lastly their is little understanding about the sequencing or timing of different forms of adaptation.

The aim of this investigation was to establish a suitable framework in which this kind of knowledge could be accessed and used by courseware designers. Not all of the necessary knowledge about adaptive design was readily available when this project started but it was believed that experience of using this form of adaptation would rapidly expand the available knowledge. In order to make this expertise available and useful to new designers, and allow it to become an expanding source of reusable knowledge, there needed to be a convenient method of storing and accessing it.

Problems of investigation

Investigation of this design support requirement was fraught with methodological difficulties, many of which are common to the design of new software development environments. The development environment was not complete enough to be used by designer subjects, and no designers had any experience of designing this kind of courseware. Also, there was no similar work that could be used to guide the content of the investigation.

We were interested in exploring how a passive case library could be used to support designers, and in particular whether it was able to guide novice designers with no experience of adaptive courseware development. The tool would necessarily need to provide learning/development support for the designer subjects as they learnt how to use a new development methodology, and a new set of tools. We were interested in observing how designer subjects used a case library, and we were interested in establishing some of the limitations of case library support for design.

Approach

The approach we used to investigate the requirements for this tool were a combination of prototyping and observational study. We set up an observational study using a crude mock-up of the design environment, then after the observation we refined the environment and tried it with the next subject. The initial mock-up used only paper- based representations for the knowledge-bases, and the case library. Once it was validated that the designer subjects could use the different forms of representation the mock-up was transferred to hypercard, plus a graphical editor. These two tools were hooked together to enable the designer to search through a library of design cases, and to copy these cases into a graphical editor and then edit, them into their main design. The indexing and search mechanisms for the case library evolved over several trials.

Findings

One of the more important decisions that was made during this study was how to represent the cases of adaptive instructional design to the designer subjects. There were several options, including, a video of the courseware being used, an executable piece of code, a graphical representation using a design notation, and a textual description. The graphical representation was

necessary as we wanted the designer subjects to actually incorporate the cases into their own graphical design. The graphical notation clearly needed supplementing in order to provide the kind of context that would make it case-like. The video and executable versions were unable to provide descriptions of learners and the rationale for the design so a textual representation was opted for. The main reservation about this approach was whether the design subjects would be able to interpret the graphical design easily enough to appreciate how the design worked. It turned out that they could in this situation because all of the design cases were relatively simple, describing a partial design that covered at most three levels of a design.

Observations of the design subjects confirmed expectations about how the cases would be used. They had several overlapping roles:

i educating the designer subject about the graphical representation, about the way the courseware should work, and about how to design this form of adaptive courseware.
ii providing a context/framework in which to think about the current sub-problem.
iii providing initial ideas about how the current sub-problem could be solved. These ideas were then adapted by the student.
iv providing nearly complete solutions for sub-problems. These needed minimal changes by the student.

The design notation encouraged a decompositional approach to the design but subjects did not generally adopt a top-down approach. As in other studies (e.g. Guindon 1990, Visser 1990) the designers were opportunisitic, sometimes being led by the cases that they found, at other times by emerging requirements The designer subjects frequently used a bottom-up approach. This has also been observed in studies of software designers, especially where they are struggling with a novel problem.

Although the cases were extensively used by all but one designer subject they did not provide a sufficiently complete level of support. A comment that was frequently made by the subjects expressed concern about the inadequacy of the case library to guide their designs. They felt that the case library needed more information about outcomes and that negative cases would have been helpful. That is, cases where learners did not learn effectively or where learners found the interaction unsatisfactory.

From the experimenter's perspective the case library did fulfil its intended role because it was used to suggest ideas about how to complete sub-tasks within the design problem, and in some cases, provided solutions that needed very little adaptation. What it failed to do was to help the subjects decide whether their adaptations of a case, and their particular use of ideas in a case, were appropriate for their problem. It could never do this because by definition a case is a specific instance of a problem solution and doesn't provide information about its generality. Also, the cases used in this study were only partial designs and they were therefore unable to take account of the full context in which they might be used. What subjects appeared to need was information about whether to proceed with their design, or not, and if not, which aspect of the design they should change. In summary, the subjects appeared to need support in evaluating the suitability of their early design ideas.

Faulty design decisions are often not recognised as such until late in to the design process, thus making correction an expensive process. The high cost of correcting mistakes is probably why experienced designers have been observed to re-evaluate their design decisions in increasingly

refined contexts (Goel & Pirolli 1992). The purpose of design evaluation is twofold, a) assessing areas that can be improved, and b) identifying areas of the design that are problematic. The latter function is seen as the most important one to support novice designers in, and is probably the more tractable of the two problems. An evaluation tool to support novice designers identify major problems with their adaptive courseware designs has been designed and partially implemented. This tool uses well established rules about adaptive courseware to identify the completeness and suitability of the adaptations that the deign will provide. It is intended to highlight gross errors not to help with detailed decisions about adaptations as these are likely to be based on the personal preferences of the designer.

Conclusions

Passive case libraries represented in a textual form can be used to support design activities. However they are limited in the extent to which they can support designers assess their designs because they are only using analogies of the actual design problem. In order to provide a sufficient support tool for novices the case library needs to be supplemented with some form of support for design evaluation. As a medium for storing and accessing design knowledge about adaptive courseware the case library seems to be adequate, although we have not investigated how designers would use it to represent new adaptive designs.

Since many applications programmers are quite narrow in the kinds of application that they build it is possible that case libraries could be used to support their design tasks. One doubt about the generality of this approach is raised by Maiden & Sutcliffe (1993) in their investigation of how design cases can be suggested to a designer. They classified data processing systems in a hierarchy, and built an expert to interrogate designers about their current problem. They found that even quite experienced designers did not prioritise the importance of system features in the same way they had. The system therefore did not suggest the most appropriate cases for the designer's problem. This work raises some doubts about building general software support tools from case libraries, although it is not clear that the same mistakes would be made if the designers were searching the case library directly rather than interacting with a tool that collects information about the current problem.

Bibliography

DISCourse (1994) DISCourse Final Report (D2008), Dornier GmbH, D-7990 Friedrichschafen, Germany.

Goel, V. and Pirolli, P. (1992) The Structure of Design Problem Spaces. *Cognitive Science*, 16, 3, 395-429.

Guindon, R. (1990) Knowledge exploited by experts during software design. *International Journal of Man-Machine Studies*, 33, 279-304.

Murray, T. (1991). Facilitating teacher participation in intelligent computer tutor design: tools and methods. Unpublished PhD thesis, Dept of Computer and Information Science, MIT.

Visser, W. (1990) More or less following a plan during design: opportunistic deviations in specification. *International Journal of Man-Machine Studies*, 33, 242-278.