Developing Courseware for Higher Education

Reflections on the Role of Evaluation

Wendy Doube¹, David Kennedy² and Juhani Tuovinen²

¹*Faculty of Information Technology* / ²*Centre for Learning and Teaching Support, Monash University, Churchill, VIC 3842, Australia*

Wendy.Doubé@infotech.monash.edu.au

Keywords: computer assisted instruction, courseware, evaluation, higher education, quality

Abstract In CFL development, relationships can be established between development team organisational maturity, selection of an iterative development methodology, and effective use of evaluation in a pedagogical as well as a technological domain, as guides to successful project outcomes. A large, higher education multimedia CFL product was developed as two projects (two years apart) by multidisciplinary teams. The first project was developed using a linear-sequential methodology. Significant technical evaluation took place but only minimal pedagogical evaluation was undertaken, mainly after alpha testing. The second project used an iterative approach to development and incorporated formative pedagogical evaluation as part of the development methodology. Resource wastage was considerably reduced and student learning requirements were better realised.

1. INTRODUCTION

The development of academic courses that use computer-based technologies in higher education has increased markedly in recent times. The use of computer-facilitated learning (CFL), particularly the development of web-based courses, is increasing as the nature of higher education itself is changing (Australian Vice Chancellors' Committee 1996). The process of developing CFL courseware, web-based courses and computer-mediated

communication (CMC) has now become an institutional point of focus rather than the domain of innovators and enthusiasts.

As CFL has moved from the fringes of higher education to being a core component of course materials, issues of software quality, student learning outcomes and integration of CFL modules within the entire curriculum, have become paramount (McNaught et al. 2000).

1.1 Evaluation in higher education

In higher education, despite summative and formative evaluation procedures being mandated as desired practice, rarely does evaluation take place during the materials development phase. The evaluation process usually concentrates on affective issues of learning – the appropriateness of teaching materials, feedback provided by the teacher, and access to resources. Formative evaluation (e.g. focus groups, peer assessment) of traditional learning activities (e.g. lecture, laboratory, class) is rarely undertaken. Only after course delivery, are assignment and exam grade distributions analysed, and student and/or peer feedback examined, so that subsequent course offerings can be amended and fine-tuned.

2. QUALITY SOFTWARE

In the IEEE Glossary of Software Engineering Terminology, software quality is defined as ‘the degree to which a system, component or process meets customer or user needs or expectations’. In actual practice, software quality means different things to different stakeholders in a software development project. The International Standards Organisation has defined characteristics of quality software in ISO9126. Project documentation in the form of a Software Quality Assurance Plan describes processes for assessing the quality of the software being developed, (e.g. inspections, walkthroughs, and code evaluations). ISO has also produced standards that address not only the quality of a product but also the quality of the processes in the development process. The Capability Maturity Model (CMM) (Humphrey 1995) is one of several models which prescribe a number of steps to continuously improve software development processes.

3. QUALITY HIGHER EDUCATION SOFTWARE

3.1 A pedagogical perspective

Strong support for the use of a constructivist perspective can be found in the literature of CFL development (Duffy and Jonassen 1991, Reeves 1997). These perspectives require that students actively interact with materials in ways that require reflection. They need to make decisions in their work which clearly reveal their own knowledge constructions. However constructivism, as an underlying pedagogy for developing courseware has been seen as problematic (cf. Merrill 1991, Anderson, Reder and Simon 1997). Dills and Romiszowski (1997) suggest that these divisions are not useful in courseware development, and regardless of the pedagogical paradigm, evaluation remains a key issue.

3.2 A development perspective

“Originally....software engineering was approached as a linear activity in which a series of sequential steps were applied in order to solve problems. Yet, linear approaches to software development run counter to the way in which most systems are actually built. In reality, complex systems evolve iteratively, even incrementally. It is for this reason that a large segment of the software engineering community is moving towards evolutionary models of software development.” (Pressman 1997, p. 832)

“In an ideal project most participants would be software engineers.” (Johnson 1997).

These statements represent opposing views of software development. While Johnson’s assertion may be perceived as naïve, it is not uncommon amongst software developers and can impact on CFL quality.

3.2.1 Identifying users and their needs

In order to meet software quality criteria, users and their needs should be defined. When the software to be developed is CFL, the wide range of possible pedagogical approaches, the lack of teacher autonomy, technical constraints and the differing requirements of stakeholders, make the definition of user needs and expectations an extremely problematic process.

Users of the final product can fall into several classifications, each with a separate set of needs to be considered. Students with more diverse

backgrounds, interests and motivation now undertake tertiary studies (Australian Vice Chancellors' Committee 1996, Bates 1997, Laurillard and Margetson 1997). Not only should the variety of learning styles displayed by students be considered, but also a variety of teaching styles. But teaching style can be very personal, and the investment cannot usually be justified unless the teacher's peers are comfortable using the product. If the product is sufficiently generic, peers and students outside the institution may wish to use it and may require additional consideration.

If we assume that the fundamental user need in CFL is learning, then specifying user needs is equivalent to specifying learning outcomes, course content, and processes which will assist students to engage effectively with the content. According to a constructivist perspective, an optimum specification will describe processes in which students will interact with the content in such a way that they will be able to construct their own meaning from it, assisted by dialogue closely resembling their face-to-face communication with a teacher.

However, developing CFL may be a new experience for the teacher who may also be ignorant of the most suitable pedagogical approach. The teacher may be unable to even visualise the final product. In this case, the specification documentation may be inadequate, and also could be subject to misinterpretation by developers. Many factors in the CFL development process, without pedagogical quality assurance, can dilute that vital teacher-student communication. Glossy, robust materials that meet software quality criteria, may be the result, but whether or not they promote learning, may be a moot point.

In order to focus on student needs and guarantee software quality criteria, appropriate formative evaluation procedures should be incorporated into the courseware specification process.

3.2.2 Selecting a model of software engineering for CFL

Most models of software engineering (SE) include similar stages, each producing similar outputs. The major differences between models lie in the order and emphasis placed on each stage, which in turn affects the iterative processes within the SE life cycle. Table 1 summarises five major SE models and some of their benefits in courseware development.

Table 1. Summary of software engineering models and their application to courseware

SE model	Benefits for courseware development
Linear-sequential (waterfall)	Strong emphasis in developing documentation which provides a point of reference and a deliverable for each stage. Requirements are documented early in the life cycle leaving developers to produce systems efficiently without interruption

SE model	Benefits for courseware development
Incremental	Educational designers and teachers gain realistic expectations of technical constraints Decreases risk of unexpected (unwanted) development outcomes.
Spiral	The specification negotiation processes of the Win Win model may be useful in large projects with many defined stakeholders. Teachers do not need to specify all of the courseware requirements prior to commencement of the programming.
Prototyping	Specific interactions in the CFL have the opportunity of student and peer evaluation.
Evolutionary	Iterative design procedures can result in a better quality product
Throwaway	Teachers can specify gradually as they become better acquainted with the system.
Evolutionary	Can take advantage of the academic year by using non-teaching weeks for workshops.
Rapid Application Development (RAD)	Includes the advantages of prototyping. Supports team understanding of the multiple perspectives held by different stakeholders.
Object-oriented	As per incremental development.

3.3 Testing and the software life cycle

“The essential feature in successful life-cycle models is quantitative forms of feedback, which make it possible for a software process to evolve, to adapt to changing environments and technologies, and to achieve the optimising level in the CMM.” (Peters and Pedrycz 2000)

Verification and validation techniques are available for each stage of the lifecycle. Testing activities are planned and documented and IEEE standards can be followed in detail. Testing in the early stages has been shown to be of the greatest benefit to a project.

One of the key features of the linear-sequential model is the provision for feedback from one stage to the previous one. In practice, this is difficult to achieve. In iterative life-cycles, feedback from testing can continue to improve a stage or set of stages until they are working well enough for the next stage to proceed.

3.4 Pedagogical evaluation

Production of traditional print-based learning materials usually follows a linear path, from teacher/author, to production, to the student, with iterations involving editing and proof-reading generally confined to the initial development process. In contrast, the development of educationally effective CFL requires a wider range of specialist skills imparted by multi-disciplinary

teams including content experts, educational designers, programmers and graphic artists (Fraser 1997). A good team dynamic with congruence of understanding of the design and development process, and good communication channels, can result in higher quality outcomes.

Other stakeholders may be even further removed from the actual teaching process. The development team may be reporting to a manager, who in turn must justify expenditure to funding bodies, either internal or external to the institution. In large teams, communication lines lengthen and the fundamental student-teacher relationship can be diluted. Responsibility for evaluation may no longer rest with the teacher. In practice, evaluation of the courseware may be neglected, often leading to less than optimal courseware development (Alexander, McKenzie and Geissinger 1998). The teacher responsible for developing the materials may move to other courses and never see the materials in use. The normal refinement processes which take place during and after course delivery may never happen.

Hedberg and Alexander (1994) have developed a model for formative and summative evaluation that addresses interface issues and student learning outcomes in the development of CFL. They argue that early formative evaluation with the target group will alleviate many potential design problems. Formative evaluation can be fundamental in CFL development if a quality product is to be delivered (Burkhardt 1992). Moonen and Schoenmaker (1992) also state that in higher education the interaction of the user with the problem is often very difficult to specify precisely, and an early prototype 'almost always elicits comments and suggestions for alterations'.

4. DRAWING ON EXPERIENCE

4.1 The first project

Two university departments – the materials development centre and the Faculty of Information Technology – received funding to develop CD-ROM based multimedia learning modules for five introductory IT subjects. An experienced multimedia project manager was employed by the materials development centre. He assembled a development team and implemented a range of formal processes within a linear-sequential development methodology. One module was produced first, and used as an example for the others, which were developed in parallel. Each module was developed in a linear-sequential life-cycle. One educational designer conducted initial student evaluations, three other educational designers worked with content

experts on the modules, and another prepared an evaluation plan to be implemented during alpha testing. The development team worked happily together in one laboratory. Content experts and educational designers regularly consulted with the team in the laboratory in an ad-hoc manner. Five modules were produced and proved popular with students. The project was cited as an example of best practice by the funding body.

However, much of the first module had to be completely redesigned and redeveloped. Two entire sub-modules were designed but never developed, while a fourth sub-module was discarded after substantial development because it failed to meet content expert expectations. After release, multiple content errors were discovered in a fifth module.

Results of the first evaluation were not translated into the design of the modules although they did influence the overall look and feel of the product. The final evaluation was never performed because of schedule over-run. After release, some ad-hoc evaluations were performed by teachers but never documented or used to improve the product.

The developers' perspective of the first product varied markedly from that of the other stakeholders. They viewed the modules that were developed with minimal academic and educational designer intervention as the most successful, even though these were the modules which required the most alteration after development or were discarded.

4.2 The second project

Two years after the first grant, a second project received funding to extend the original product. A content expert (first author) who had been involved with three of the original five modules was appointed as academic project leader. Two educational designers, new to the project, worked on the project, one to supervise design, the other evaluation. By now, the original technical development team was busy with other projects and had reached a control and co-ordination level of maturity. It could no longer function effectively with the constant interruptions of unscheduled visitors and implemented firmer standards and controls, including time logs, in order to manage multiple projects simultaneously.

The academic project leader felt that the problems with the first module of the first project were the result of the content expert being divorced from the development process between design and delivery. She had therefore work-shopped with the development team throughout the life cycle of other modules. In the second project it was felt that the development team was not equipped to deal with a prototyping methodology, and a single additional iteration was introduced into the development life cycle. A 'broad design' stage, which produced user interfaces and non-standard interactions to assist

content experts define their needs, preceded a 'detailed design stage' from which the final product would result.

The educational designers drew up an evaluation plan which was integrated into the development methodology. Task analysis, and overall course perspectives were obtained from students before the project started and fed into the broad design. Peers and students evaluated storyboards, the implementations of first the broad and then the detailed design which was edited and then released. On-going evaluation was planned post-release.

In the second project, all designed modules were fully developed. Academics were more satisfied with the application of the pedagogical design perspective in the resultant specifications. However, the development team's new procedures assumed linear-sequential development and required initial detailed user specifications to remain largely static throughout the life cycle. Formal communication procedures impeded the clarification resulting from informal contact between technical developers and other team members during the broad design stage. Some conflict was inevitable. In compensation, weekly multidisciplinary workshops were scheduled for the detailed design.

4.3 Technical vs. learning evaluation

Both project teams included experienced educational designers, well versed in evaluating the effect of materials on *learning*. They also included academics, used to continually refining their materials in response to qualitative and quantitative student feedback.

However, in the first project, when new to CFL, academics and educational designers had less influence on evaluation processes:

- following the traditional waterfall model of software development, learning evaluation was scheduled after development, during software testing
- the focus of this evaluation became software validation and verification, rather than learning.
- the impact of the user interface on learning was still being explored. Issues such as the impact of spatial arrangements and navigation on learning were addressed in the only educational design evaluation actually executed
- any post-development educational design changes resulting from evaluation would have been too costly to implement. In any case, by that time, all project funds were spent.
- any post-development evaluation would only impact on future products and therefore was perceived as a reduced priority. After release of the

product, academics responsible for it had moved on. Often evaluation didn't take place at all.

In summary, we could infer that in the first project, evaluation was driven by more *technological* rather than learning requirements. The scant pedagogical evaluation concentrated on interface and navigation issues.

In the second project:

- students and academics had become more familiar with the CFL interfaces which became less of an evaluation priority
- content experts and educational designers were more involved in determining the development process which promoted the evaluation of learning
- the value of iterative software development methodologies for CFL was now apparent. These methodologies created more opportunities for evaluation during the lifecycle
- storyboarding and scripting techniques had become sophisticated enough to be able to provide a good indication of the final product. Some evaluation was able to take place before development

In summary, we could infer that as CFL became more widespread, and as the project team gained a greater understanding of courseware development processes, evaluation concentrated more on *learning* and less on technological issues.

5. CONCLUSION

The experience of these two large CFL development projects highlighted the importance of selection of an iterative model of software engineering for courseware development. In the first project, a linear-sequential model reduced the impact of pedagogical evaluation on the specification of user requirements, except in issues of interface design. The major focus of evaluation was linked to software testing by which time it was too late to change the specifications. As a result, design and development effort was wasted on modules that did not meet user requirements. A more iterative model of SE was adopted in the second project, where formative peer and student evaluations were integrated into the SE life cycle. Very little effort was wasted and the product was more congruent with the educational intentions of its authors.

Inferences may be made that CFL development will be enhanced by

- development team maturity,
- selection of an iterative SE model
- equal emphasis on learning and technical quality control

- continuous involvement of content experts and instructional designers throughout the life-cycle.

REFERENCES

- Alexander, S., McKenzie, J. and Geissinger, H. (1998) *An evaluation of information technology projects for university learning: Executive summary*, Report, Committee for University Teaching and Staff Development (CUTSD).
[http://www.autc.gov.au/in/in_pu_cu_ex.htm] 26.10.2000.
- Anderson, J. R., Reder, L. M. and Simon, H. A. (1997) *Applications and misinterpretations of cognitive psychology to mathematics education*.
[http://act.psy.cmu.edu/ACT/papers/misapplied-abs-ja.html] 1998, February.
- Australian Vice Chancellors' Committee (1996, 9 October) *Exploiting information technology in higher education: An issues paper*. [http://www.avcc.edu.au/avcc/pubs/eitihe.htm] 10.02.1997.
- Bates, T. (1997, 18-20 June) Restructuring the university for technological change. In *Carnegie Foundation for the Advancement of Teaching: What kind of university?* Proceeding of the Carnegie Foundation for the Advancement of Teaching Conference, London, pp. 1-23.
- Burkhardt, H. (1992) Classroom observation in courseware development. *International Journal of Educational Research*, 17, pp. 87-98.
- Dills, C. R. and Romiszowski, A. J. (1997) The instructional development paradigm: An introduction. In *Instructional Development Paradigms*, C. R. Dills and A. J. Romiszowski (eds), Educational Technology Publications, Englewood Cliffs, New Jersey, pp. ix-xxx1.
- Duffy, T. M. and Jonassen, D. H. (1991) Constructivism: New implications for instructional technology? *Educational Technology*, 31 (5), pp. 7-12.
- Fraser, K. (1997) Academic developers. Enhancing the pedagogical value of multimedia products developed by higher education learning environments. *International Journal of Academic Development*.
- Hedberg, J. and Alexander, S. (1994, 28 June – 2 July) Implementation and evaluation: The forgotten end of interactive multimedia development. In *APITITE 94*, M. Ryan (ed), Proceedings of the Asia Pacific Information Technology in Training and Education Conference and Exhibition, APITITE 94 Council, Brisbane.
- Humphrey, W. S. (1995) *The systems engineering Capability Maturity Model: A discipline for software engineering*. Addison-Wesley.
- Johnson, L. (1997) *Software Engineering 3A, 433-341: First Semester Handbook and Notes*. The University of Melbourne Department of Computer Science.
- Kennedy, D. M., Fritze, P. and McTigue, P. (1997, December 7-10) An interactive graphing tool: The meeting of pedagogy and technology. In *What works and why, ASCILITE '97*, R. Kevill, R. Oliver and R. Phillips (eds.), Proceedings of the Australian Society for Computers in Learning in Tertiary Education, Perth, pp. 331-337.
- Laurillard, D. and Margetson, D. (1997) *Introducing a flexible learning methodology: Discussion paper (7)*. Griffith University, Brisbane.
- McNaught, C., Phillips, R., Rossiter, D. and Winn, J. (2000) *Developing a Framework for a Useable and Useful Inventory of Computer-facilitated Learning and Support Materials in Australian Universities*, (DETYA No 6472 HERCOA), Canberra.

- Merrill, D. M. (1991) Constructivism and instructional design. *Educational Technology*, 31 (9), pp. 45-53.
- Moonen, J. and Schoenmaker, J. (1992) Evolution of courseware development methodology: Recent issues. *International Journal of Educational Research*, 17, pp. 109-121.
- Peters, J. and Pedrycz, W. (2000) *Software Engineering: An Engineering Approach*. Wiley, New York.
- Pressman, R. S. (1997) *Software engineering: A practitioner's approach*. 4th ed., McGraw-Hill, New York.
- Reeves, T. C. (1997) Established and emerging evaluation paradigms for instructional design. In *Instructional Development Paradigms*, C. R. Dills and A. J. Romiszowski (eds.), Educational Technology Publications, Englewood Cliffs, New Jersey, pp. 163-178.
- Robertson, L. A. (2000) *Simple Program Design*. 3rd ed., Nelson, Australia.

BIOGRAPHY

Wendy Doubé has taught computer programming, software engineering and computer graphics to both on and off-campus students, producing course materials in a wide range of media, including the multimedia modules discussed in this paper and chapters on object-oriented design in the textbook, 'Simple Program Design' (Robertson 2000). Her publications have explored the impact of distance and media on teaching and learning computing. She is currently investigating the motivational impact of multimedia interactions in learning.

David Kennedy is an educational designer who works with academic staff across the university to develop more flexible learning environments that require the appropriate use of technology.

Dr. Juhani Tuovinen is the Senior Research Fellow in Interactive Multimedia at Monash University's Centre for Learning and Teaching Support. He has been a secondary school teacher and a university lecturer in education. His expertise is in distance education, the use of computer and communications technology in education and cognitive science. His current research interests include the application of cognitive principles to distance and flexible learning and the use of modern technology in university education.