The CourseMaster CBA System: Improvements over Ceilidh

Eric Foxley, Colin Higgins,
Tarek Hegazy, Pavlos Symeonidis and
Athanasios Tsintsifas

The CourseMaster CBA System: Improvements over Ceilidh

Eric Foxley
Colin Higgins
Tarek Hegazy,
Pavlos Symeonidis
Athanasios Tsintsifas

Learning Technology Research (LTR)
School of Computer Science and I.T.
The University of Nottingham
Jubilee Campus
NG8 1BB
United Kingdom
ltr@cs.nott.ac.uk

Abstract

While teaching in all its forms can sometimes be fun, for most people marking student work is tedious, boring and in general hard work. A courseware system is presented that not only provides on-line support for courses, but importantly can automatically assess student work.

CourseMaster can mark several types of coursework in a non-trivial manner. That is, criteria can be set against which the work is thoroughly assessed. Specifically, the system is particularly effective at marking computer programs in several languages including Java and C++. It can also mark diagrams and assist in the marking of essays.

In addition to marking, the system also supports the provision of lecture notes and web pages and links. It can be used to collect any on-line work and enforce deadlines. Finally, it provides a suite of web- based tools that allow the easy management of courses. CourseMaster has been used "live" at Nottingham with great success for three years and is based on the earlier Ceilidh system which was developed and used over the preceding ten years.

Keywords

Courseware; automatic marking; automatic grading; CBA; summative assessment; automatic assessment of programming courses.

Introduction

A courseware system is presented which facilitates teaching and marking and enhances the student's learning experience. Courseware systems typically consist of a suite of material from which students can learn and a delivery mechanism for providing this material and for administering the running of courses. The material presented to the students might consist of lecture notes (in a variety of formats), tips and guides, messages to the students (e.g. deadlines and assignment dates), diagrams, animations of algorithms etc. and, of course, exercise questions and assignments to be attempted. Administration requires the collection of registers and assignments, monitoring student and cohort progress, giving feedback and reporting marks etc. Ideally, courseware also includes tools to assist in the assessment of the course; perhaps the most important aspect and certainly the least readily available.

Teaching can be fun! Administration can be delegated. However, marking is monotonously repetitive and assessment in general is probably the least liked task of most academics! Hence, whilst the material, delivery and administration systems are an important aspect of courseware (especially in distance learning environments), running a course can be done without them. Many other courseware systems now exist for the production and dissemination of course material to students. Ceilidh pioneered a crucial aspect of courseware, that of *automatic assessment*. Although the Learning Technology Research group (LTR) is involved in the material and administration aspects of courseware which are highly important in providing a stimulating and enriched learning environment and in supporting the teaching of a course, the main concern is with automatic assessment of student work, and in particular the marking of student programs, diagrams and essays.

The LTR group at the University of Nottingham has been working in the area of automatic assessment for thirteen years. This work has resulted in many tools and papers, the major products of this work being the Ceilidh system and more recently the Ceilidh CourseMaster System, referred to as CourseMaster (CM) for short. Importantly, these developments have been used to teach active courses to students from the outset, resulting in practical, tested, working solutions. The feedback from this use has resulted in improved systems incorporating valuable experience, particularly of automatic assessment.

This paper first describes and reviews the old Ceilidh system and how the new software, CM, builds on the experience gained from it. A description of the CM system follows, including its functionality, design philosophy and usage. Finally our experience of using CM for the last three years is described and conclusions are drawn.

The Ceilidh System

Ceilidh is a courseware system implemented with a collection of programs and tools written in a variety of languages including shell, C and awk, which execute in a Unix

environment. The first version was used in 1988, and as a result of the continuous incorporation of enhancements suggested by users over the years it has become increasingly difficult to understand, maintain and support.

The three courseware dimensions (presentation of material, administration and assessment) are provided in a variety of ways. Material is provided as a set of text files and web pages, administration is enabled via active web pages and assessment can take a variety of forms. The most powerful form of automated assessment is the automatic marking of programs in a variety of programming languages, a unique feature of Ceilidh, given the depth to which it can mark. In addition, Ceilidh can mark multiple choice questionnaires and can be used to collect any on-line coursework. A set of tools gives a human marker grammatical and spelling measures to assist in the hand marking of essays.

Five roles of users are represented within the system, with each role having progressively more access rights and authority. Firstly, there are the students who can browse notes, read questions, look at the message of the day, “setup” a solution (download a skeleton file etc.) and submit a solution. Solution development is performed externally to the system via a Programmer’s Development Environment (emacs in our case). Secondly, there are tutors/lab assistants who in addition can look at model solutions, read tutor help files and monitor any student’s progress. Thirdly, there are teachers of a course who in addition can chose which exercises to use and when to set deadlines. Fourthly, course developers can build exercises with their detailed mark schemes and enter new material. Finally, there is the system administrator role where new users or courses are added and general maintenance of the system performed. Numerous other rights and responsibilities are allocated to each of these roles as well as those mentioned here.

Ceilidh was used widely with courses for around 15 different programming languages. While successful, it had several disadvantages. For instance, it was based around the Unix operating system and required knowledgeable system staff to install and maintain. The assessment mechanism was powerful, but feedback to the students was limited. The user interface was for many years based on ASCII character terminals.

CourseMaster – a better Ceilidh

Because of the nature of Ceilidh’s on-line development, as time progressed and the functionality was enhanced, it became more unwieldy and harder to support and extend. Finally, in 1997 the redevelopment of Ceilidh commenced, from the beginning, to rigorous software engineering standards, ensuring its continued use and ease of support. To this end, the system was redesigned using object-oriented methods and re-implemented to give the new system which was renamed CourseMaster (CM).

Taking into consideration Ceilidh's success, all parts that worked well were identified and redesigned while aiming for better performance, ease of maintenance and increased extensibility and usability. As table 1 illustrates CourseMaster preserved most of Ceilidh's functionality and added more. Most notable are the options for viewing an analytical tree of the exercise's assessment. This is usually accompanied with extensive feedback. Other new options allow the students to configure their work environments to match their preferences.

System levels / User types	System	Course / Unit	Exercise
Student	⊗ - View system documents. ○ - Set environment properties. ○ - Customize system view. - View system MOTD*. ⊗ - Register to a course. ×	⊗ - View course documents - View course notes. ⊗ - View course summary. ⊗ - View course MOTD. ⊗ - View total course work. ⊗ - View all student's marks. - View unit documents. ⊗ - View unit notes. ⊗ - View unit summary.	⊗ - Set up exercise. ⊗ - View exercise question. ⊗ - View exercise skeleton. ⊗ - Develop exercise. ⊗ - Submit exercise. ○ - View analytical mark and feedback ⊗ - View exercise test data (TD). ⊗ - View marks. - Execute solution: ⊗ - Interactively. ⊗ - With TD. × - With teacher's TD. × - Execute teacher's solution with student TD. ⊗ - Enquire by e-mail.
Tutor		⊗ - View students' course marks. ⊗ - View Student Statistics. ⊗ - Check course state. × - Register student to a course. ⊗ - View student's work.	⊗ - View student solutions. × - Mark solutions (manually or automatically). ⊗ - View statistics. ⊗ - View missing / submitted students. ⊗ - View tutor help. ⊗ - View plagiarism results. × - Find tutees for a specific tutor.
Teacher	⊗ - Add / Delete students, tutors, teachers. × - List tutor's tutees. × - View audit trails.	⊗ - Edit course MOTD. ⊗ - Check / Edit course properties. ⊗ - Edit unit properties.	⊗ - Edit Tutor help. ⊗ - Edit exercise question. ⊗ - Change exercise properties. ⊗ - Expunge student's work. ⊗ - Set borderline. ⊗ - Search for Plagiarism.
Developer		⊗ - Create / Add courses. ⊗ - Create / Add units.	○ - Copy / Delete exercises. × - Author exercises.
Administrator	⊗ - Edit system MOTD. ⊗ - Add / Delete Administrators. ○ - View error log. ⊗ - Register students.	⊗ - Edit course documents. ⊗ - Install new courses.	○ - Give extensions. ○ - Copy / Delete exercises. × - Author exercises.

×: Ceilidh ○: CourseMaster ⊗: Both Ceilidh and CourseMaster * MOTD: Message of the day.

Table 1: Users' responsibilities at different levels of system use – Ceilidh vs. CourseMaster.

Object-oriented methods, and theory from frameworks and design patterns, were used in the design of CM so that benefits could accrue from such areas as extensibility, and maintainability. Java has been use to implement CourseMaster in order to satisfy the deployment requirements arising from the wide range of computing configurations that exist in academic institutions. Java facilitates platform independence and offers a convenient distribution mechanism with its Remote Method Invocation (RMI) mechanism. RMI is considerably more effective than other client/server methods and relatively easy to use as it makes the network transparent to the programmer. The latter

feature has been especially sought as the early stages of development identified the inherent complexities of building distributed systems in heterogeneous environments.

In an effort to re-organise Ceilidh's functionality in a more flexible way, we identified the dependencies, commonalities and variations between Ceilidh's different parts. We then abstracted the commonalities into class hierarchies, defined explicit point of extensions and parameterisation for all the variations and decreased the dependencies by separating the various responsibilities between seven logical parts. We implemented those parts as servers that operate as remote objects using Java's RMI. Figure 1 depicts a high level overview of the relationship between users and those seven parts. Table 1 enlists the responsibilities of each of the servers.

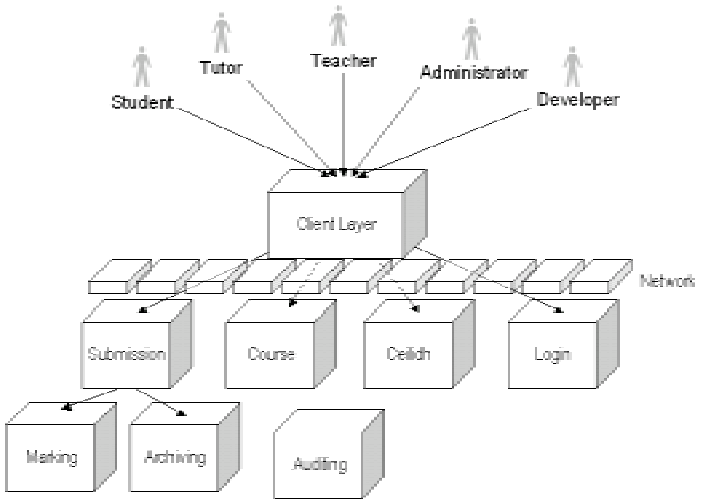
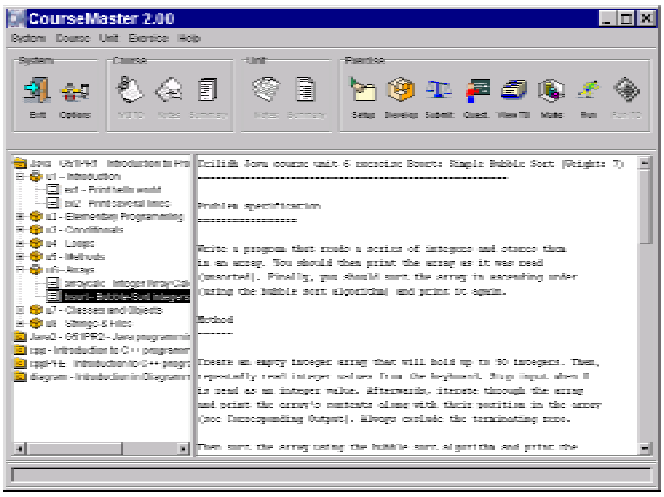


Figure 1: A high level view of CourseMaster main parts

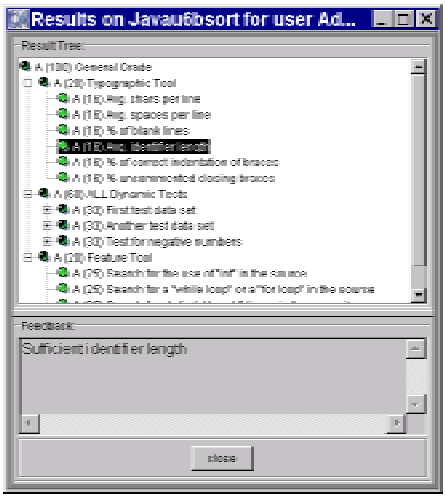
Server Name	**Function**
`Login`	Authorises users to perform the requested tasks
`Course`	Manages course material and responds to requests
`Submission`	Decides between accepting and rejecting a submission.
`Marking`	Marks the submission and produces analytical feedback
`Archiving`	Archives student work and issues unique receipts
`Auditing`	Logs audit trails for various configurations
`Ceilidh`	Provides course related information to its clients

Table 2: CourseMaster's servers and their responsibilities

During redevelopment many enhancements were made. Most notably are: the development of a much enhanced GUI client (figure 2a) for students making the system much easier to use; the greatly enhanced feedback mechanism (figure 2b) enhancing the learning experience; and the addition of a powerful new subsystem for marking that is highly extensible. In theory, the marking system can be set to mark any type of document as long as correct metrics can be identified and developed. This is demonstrated by the capability of the system to mark successfully exercises in circuit (figure 3b) and object oriented design (figure 3c).

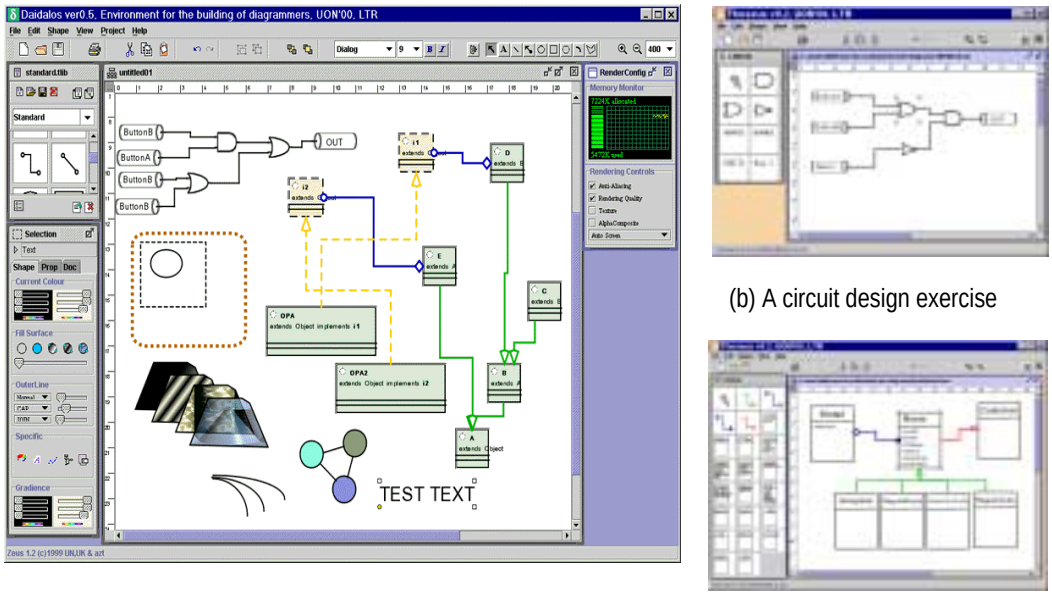


(a) The student view for a course



(b) Exercise results for a student

Figure 2: CourseMaster's student interface



(a) The authoring environment for diagram-based exercises

(b) A circuit design exercise

(c) An OO design exercise

Figure 3: CourseMaster's diagram-based assessment

The student level functionality is provided by the set of the seven interlinked servers, while three Java clients provide the student user interfaces (see figure 2 for the version). Student feedback is shown in figure 2(b). The tutor, teacher, developer and administrator facilities are provided by an extensive set of web pages that utilise CGI scripts and Unix software tools such as shell, awk, grep etc. (figure 3). For Windows environments these software tools are compatible with a set of Unix look-alike tools provided for in the Cygnus package. In general CM is a great improvement over Ceilidh in terms of usability, maintainability, expandability and feedback.

Evaluation

Copies of Ceilidh were taken by over 330 institutions around the world. Usage is more difficult to gauge as the original system was freely available and not every user bothered to register. Given the numbers that did either register, email or enquired for upgrades, it is estimated that the number of sites using Ceilidh was about 100, the biggest of which, used it on approximately 2300 students per year.

The new CM system has been in active use by Nottingham, concurrently with development, for three years. There were 291 students using the system on the latest Java module during the year 2000/2001. Student feedback is extremely positive with the number of emails requiring assistance reduced by a factor of about ten compared to

Ceilidh emails from previous years, despite the number of students more than doubling. Complaints were usually from students who had coded before and overlooked the stricter constraints placed on their style by the software engineering practices that CM enforced.

It is noticeable that female, overseas and mature students in general preferred CM. The strongest complaint was due to the scaling process applied to marks at the end of the year; this despite several warnings to the students that this would take place. Because students have multiple submissions with detailed feedback it is not surprising that the average mark for the course is very high (around 92% this year). Due to compensation rules, marks with this average are not acceptable by the School and so the marks are scaled down (via the administration sub-system). Even with this problem, the number of complaints was in single figures, and after detailed explanation eventually all but one student was satisfied with the process.

Early beta versions of the system have also been tested at Ngee Ann Polytechnic (Singapore) and by Kings College London (UK). These have been used for C and C++ courses respectively. Feedback from academics has been positive with both institutions able to run pilot schemes on real students and to set their own exercises and mark schemes. A beta of the system was made available in December 1999 and has been taken by nearly 20 institutions. The first full release was in June, 2000. Currently about 15 institutions have bought CM including sites in Ireland, Singapore and Australia.

Conclusions

Ceilidh was used successfully for over ten years at approximately 100 sites world-wide, marking work on many programming languages. The new CourseMaster system has been running for three years at the University of Nottingham covering an introduction to Java course and an intermediate Java course each year. Several other Universities have also run at least one Java course. It has also been piloted with C and C++ courses at other academic establishments. Most recently, the generic diagramming authoring environment (figure 3a) has been integrated and used to mark both object-oriented and electronic circuit diagrams. Student reaction is overwhelmingly positive with a few minor exceptions. Staff acceptance is also high, especially given that most staff no longer need to mark programming course-works for tutorials. However, staff can still keep track of their tutees via the administration sub-system.

At Nottingham some students have used both Ceilidh and CM depending on their year of entry and courses selected. Students definitely prefer the use of CM over Ceilidh, showing that not only is the new system easier to administer, but also has a better interface and gives more informative feedback to students. Enhancements are in hand to improve the learning experience of students while reducing the burden on teaching and administrative staff.

CM improvements in hand or planned include:

- the implementation of submission over the web (to allow the easy integration of CM marking with other courseware and material presentation environments);
- the production of a remote console (allowing CM to be administered remotely and from any number of sites);
- automatic emailing of student progress to tutors;
- tools for marking the object-oriented aspects of programs in greater depth.

In addition the LTR group is researching into many longer term items. These are all based around helping students whist minimising staff resources and include web based and AI based systems. For instance, agent based systems to monitor students progress and automatically suggest when they are able enough to move on to new topics or where they need to go and what they need to read to cure specific problems. These problems can be identified by a variety of mechanisms such as monitoring CM feedback or course material access or more directly by parsing emails or other requests for help.

Ceildih's successor has proven to be invaluable to the University of Nottingham and to other academic institutions around the globe. CourseMaster demonstrates considerable improvements over Ceilidh. Its architecture and implementation satisfy the objectives of maintaining the old functionality while increasing performance, scalability, maintainability, extensibility and usability. The changes made on the assessment and administration processes have been successful too, as they improved the expressiveness of the marking and the management of courses making automatic assessment more approachable to academic institutions.

References

The following Ceilidh and CourseMaster related documents (distributed with Ceilidh [1] or CourseMaster [2], and available over the web) are recommended as initial reading material: the Student Guide[16] , the Tutor Guide[17] , the Teacher Guide[18] , the Developer Guide[19]

A number of journal papers highlight the major aspects of our work in Ceilidh: from the Software Quality Journal[11] , the Association for Learning Technology Journal[12] and the Journal Education Technology[13]

Finally some of many conference papers summarise the various aspects of Ceilidh: The CBLIS conferences at Kiev[14] , Dublin[15] , Amsterdam[7] ,Twente [6], Vienna[8] and[9] , York[10], Loughborough[4]

1. http://www.cs.nott.ac.uk/~ceilidh/papers.html

2. http://www.cs.nott.ac.uk/CourseMaster/
3. E Foxley, C A Higgins, A Tsintsifas, ‘Security Issues under Ceilidh’s WWW Interface’, *Proc ICCE’98*, Vol 1 pp235-240, Beijing, China, 14-17 Oct, 1998.
4. E Foxley, C A Higgins, A Tsintsifas, ‘The Ceilidh System: A general overview’, *Second Annual Computer Assisted Assessment Conf*, Loughborough, UK, 17-18 June 1998.CourseMaster General
5. E Foxley, C A Higgins, A Tsintsifas, P Symeonidis, “The CeilidhCourseMaster System, An Introduction”, 4th Java in the Curriculum Conference, South Bank University, UK, 24th Jan 2000.
6. E Foxley, A Tsintsifas, C A Higgins, P Symeonidis, Ceilidh, a system for the Automatic Evaluation of Students Programming Work, CBLISS 99, University of Twente, Holland, 2-7 July, 1999.
7. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Modh Zin, Integrating Software Quality Assurance into the Teaching of Programming, Amsterdam, Proceedings of the CSR 10th Annual Workshop "Applications of Software Metrics and Quality Assurance in Industry", Amsterdam, 1993.
8. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Mohd Zin, Ceilidh: A course administration and marking system, Proceedings of the International Conference on Computer Based Learning in Science, Vienna, December 1993.
9. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Mohd Zin, Experiences with the Ceilidh System, Proceedings of the International Conference on Computer Based Learning in Science, Vienna, December 1993.
10. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Mohd Zin, CEILIDH - a management and marking system, Assessment in Higher Education Computing, York, June 1993.
11. Steve Benford, Edmund Burke, and Eric Foxley, "A System to Teach Programming in a Quality Controlled Environment", The Software Quality Journal 2, pp.177-197 (1993).
12. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Modh Zin, "Early experiences of computer aided assessment and administration when teaching computer programming", Association for Learning Technology Journal 1(2), pp.55-70 (1993).
13. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Modh Zin, "Ceilidh as a Course Management Support System", Journal of Educational Technology Systems 22(3) (September 1993).
14. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Modh Zin, The Ceilidh Courseware System, Proceedings of the International Conference on Computer Technologies in Education 1993, Kiev, Ukraine, September 1993.
15. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Modh Zin, Experience using the Ceilidh System, Proceedings of the All

Ireland Conference on delivering the Computer Curriculum, Dublin, September 1993.

16. Steve Benford, Edmund Burke, and Eric Foxley, Student's Guide to the Ceilidh System (2.4), LTR Report, Computer Science Dept, Nottingham University, 1995.
17. Steve Benford, Edmund Burke, Eric Foxley, N H Gutteridge, and A Mohd Zin, Tutor's Guide to the Ceilidh System (2.4), LTR Report, Computer Science Dept, Nottingham University, 1995.
18. Steve Benford, Edmund Burke, and Eric Foxley, Teacher's Guide to the Ceilidh System 2.4, LTR Report, Computer Science Dept, Nottingham University, 1995.
19. Steve Benford, Edmund Burke, and Eric Foxley, Course Developer's Guide to the Ceilidh System, LTR Report, Computer Science Dept, Nottingham University, January 1994.