An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers

Helen Sharp and Pat Hall

Computing Department, The Open University, Walton Hall, Milton Keynes MK7 6AA, UK,
+44 1908 653638
http://mcs.open.ac.uk/hcs2/

Abstract: The Open University's *M880 Software Engineering* is a postgraduate distance education course aimed at software professionals. About half of the course is taken up with standard teaching materials, and about half involves case studies which complement the standard materials. The case study element of the course (approximately 80 hours of study) explores various aspects of software engineering and is presented through an innovative interactive multimedia simulation of a software house Open Software Solutions (OSS). The student 'joins' OSS as an employee and performs various tasks as a member of the company's project teams. In this paper, we present the background to the development, describe the multimedia and its use by students, and present the results of two evaluation efforts to date.

Keywords: Educational multimedia, distance education, simulation

1. Introduction

The Open University educates mature part-time students at a distance. Courses are delivered through a mixture of media such as text, software, video, television programmes and audio tapes, and increasingly the Internet.

Software engineering courses at university are concerned with the principles underpinning the development of large software systems in real industrial settings. Contact with real software projects is therefore important. Up to 1997 our M860 Software Engineering course included a video of a software development project, but software engineering is a highly dynamic area and the materials, including this video needed replacing. We began planning our replacement course, M880, in 1996 and at this time interactive multimedia was becoming popular; we saw it as an ideal opportunity to explore whether we could use this approach to give real project experience that enhanced our students' learning.

1.1 Why multimedia?

Ideally we would like our students to all work in identical industrial projects in which we could control the experiences through which they would learn and obtain motivation for their learning. This is situated (McLellan, 1995) or anchored (Bransford et al, 1990) learning. Case studies described in text or on video can be engaging, but they cannot approach this real experience. Multimedia held out the prospect of doing this. In addition, our students have a lot of text-based material to read, and multimedia would provide a change from reading text.

Educational multimedia products vary considerably in the facilities and sophistication of interaction they provide (Boyle, 1997; Ed-Media, 1999). At its simplest, interaction takes the form of 'page-turning', moving between screens full of text using some method such as a button. Page turning may be slightly extended by using hypertext links. Animations may permit the visualisation of dynamic processes, though all too often it is only possible to set the animation running, and then watch, as with the internal combustion engine in Encarta 97 Deluxe.

We were looking for more sophisticated interaction styles such as ‘conceptual interaction’ (Laurillard, 1996), where users do things at a level meaningful for their learning use. We wanted experiential learning, with students making choices as they would have to within a real project, and then seeing the consequences of those choices.

This led us to design the simulation of a software house called Open Software Solutions (OSS). The course case studies are represented by projects being run by OSS staff and the student ‘joins’ the company as a new employee and then participates in the work of the projects. The tasks set by the project manager for each case study represent exercises to practice the application of techniques and principles learned in the standard teaching materials.

1.2 M880 and the CCI Programme

M880 Software Engineering is a 240 study-hour (UK 30-credit points) course studied over 6 months. It forms part of the Computing for Commerce and Industry (CCI) Postgraduate Diploma and associated MSc offered by the Computing Department.

The CCI programme is aimed at software professionals who continue with full-time employment while undertaking a postgraduate course of study. Students are sent a pack of course materials to study on their own at their own pace. Each student is assigned to a tutor, who offers support and guidance via telephone or email contact (each tutor is allocated around 20 students). Learning is paced through four assignments marked by the tutor according to detailed mark schemes provided by the M880 course team.

For M880 students the pack of course materials contains:

1. Foundation material: a standard and mature textbook (Pressman, 1994) with ‘wrap-around’ material to complement and deepen the treatment of topics in the textbook. This material includes small exercises to allow students to practice techniques and check their understanding (110 hours study)
2. Case studies: provide further practice of techniques and principles, contextualised within a larger development context. The case studies, and feedback on the exercises students are asked to complete, are presented through the Open Software Solutions (OSS) multimedia environment. This environment is the main focus of this paper. (80 hours study)
3. Four assignments (40 hours work)
4. Study guide material: general information and a study calendar to help with time management and deadlines. (10 hours study)

Students learn about techniques and principles, and acquire basic knowledge about them by studying the foundation material. The case studies give further, more challenging applications of the knowledge and skills. Study of the two aspects of the course – standard teaching material and case studies – is interwoven so that the case study illustrating a topic is studied just after the basic knowledge has been acquired.

The course itself covers 12 software engineering topics: The Nature of Software, Software Quality Management, Software Development Lifecycles, Requirements Specification, Software Design, Testing, Maintenance, Software Reuse, Planning and Management, Human Factors, Professional Issues, and Software Process Review.

2. Case Study Design

A key question for the design of the environment which we asked ourselves regularly was whether the multimedia was giving us something better or different than other (single) media would have done. With every design point we considered, we wanted to be sure that multimedia was the right presentation for the material.

2.1 Design considerations

From the predecessor course, we knew that our students had trouble with the following aspects of software engineering:

1. the character and the significance of human interaction within a software development project;
2. the size and complexity of most software projects;
3. the relationships between different notations and techniques, and how they can be used together.

To address these aspects, we therefore wanted our set of case studies to include:

1. an organisation with a history, a culture, personnel and working projects, to give a realistic context including some aspects of human interaction
2. one project which is much too big for any software engineer or student to understand all of it (as so frequently happens in practice);
3. one project which is manageable, and which would allow a student to take part in most activities so that they could see across development.

In addition, we wanted to include:

4. a reasonable spread of different application types so that the case studies would cover the range of different techniques taught in the standard teaching materials.
5. a variety of media as appropriate to the case study, and also to engage the students, e.g. video, audio, software prototype, presentation slides and software simulation.
6. OSS personnel to reflect the wide variety of ethnic backgrounds of our students, and to include both genders.
7. projects that involved different roles within OSS as well as clients external to OSS, because a software development project includes various stakeholders and there are often tensions between the different roles and we wanted to illustrate this.

In a real software house, employees would be able to go to any project within a company and talk to the project team members; at the same time, they would be told by their ‘boss’ which project they had to work on. In this same spirit, we decided to allow students free access to the projects, unconstrained in the order in which to address the project tasks; at the same time, the study calendar directed them to the appropriate project for practicing the current topic. This allowed students to repeat the case study tasks without complicated history tracking. To reinforce the project guidance, we introduced a manager for each project to tell students in which order to approach the tasks. We thereby hoped to achieve a good balance between freedom and focus.

For educational purposes, students require formative feedback. One of our aims was to provide uniform project experience to all students, and at a distance we cannot expect all our tutors to offer exactly the same advice, answer queries in the same way, provide comparable hints and so on. So we decided to offer feedback as sample solutions plus explanations from the project manager as if part of the normal process of staff development. The disadvantage is that students do not receive tailored feedback through the environment, but we were happy for tutors to discuss the students' own answers in the context of the sample solutions.

In the spirit of conceptual interaction, tasks needed to be meaningful in the context of the students' learning, i.e. they had to reflect the topics being taught through the standard teaching materials, and therefore needed to build upon their acquisition of basic knowledge. The tasks had to be challenging and set in the context of realistically complex software projects, but as it is an educational exercise, the tasks must also be manageable and reasonably self-contained. This led us to design a variety of student tasks, most of which address only one or two aspects of each project. We considered a variety of company structures and projects, but eventually settled on the case study elements described below. A more detailed explanation of the student tasks is given in Section 4.

2.2 Open Software Solutions

Open Software Solutions (OSS) is a small company founded in 1978 to exploit the software development skills of its four founders. Since the company was formed, they have been involved in developing a variety of systems including those with real-time, database management and transaction processing elements. OSS provides the organisational context for our case studies and is currently involved in four projects:

1. CIRCE (Corporate and Individual Records for Customers and Enquirers) is an information systems development project at The Open University to replace the existing disparate course development and student support management systems (around 54 of them). The CIRCE project involves many teams of developers and several hundred users. OSS assisted the Open University in developing their own approach to software development and the company has now been asked to help evaluate the approach. The OSS project team consists of the manager, Bill Haley, and the student. This M880 case study is based on a live project and reflects the people, roles, issues, opinions and software development method concerns that were faced by this project team. It fulfils the requirement of a project that is too large and complex for a single software engineer or student to understand it all.
2. Production Cell. This project involves software to control an industrial production cell in a metal-processing factory. This is a real-time domain with safety properties. OSS is using the project to train its staff in object-oriented techniques and formal methods (i.e. the Z notation). The OSS project team consists of the manager, Tricia Bailey, and the student. This M880 case study is based on a standard problem in computer science teaching and research used to illustrate the use of formal methods.
3. SummerSun is a fictitious travel agency. OSS is developing a system for SummerSun that will enable customers to select package holidays from a range of holidays that SummerSun offers. The short-term aim of the project is to deliver this system, but there is also a longer-term aim which is to sell similar systems to other travel companies. The approach being taken to the development is a conventional database approach in the structured analysis and design style, starting with a short requirements gathering stage and using an OSS 'in-house' life-cycle model. The OSS project team consists of the manager Aswin Nakhwa, a programmer Jo Richards, and the student. This M880 case study was developed from an amalgamation of experiences from real projects, but does not reflect any one in particular. This project has been used previously in other courses, although most of the materials in the OSS case study were developed specifically for M880.
4. Quality certification project has two aims: in the short term to achieve ISO 9001 and TickIT registration for OSS; and in the medium to long term, to use the SEI's Capability Maturity Model as the basis for process improvements in OSS. The OSS project team consists of OSS's technical director and project manager, Bronwen Davies, and the student. This M880 case study was based on the consultancy work of one of the course advisors.

3. Interface Design for the OSS Environment

We wanted the system to be as engaging as possible, and considered a wide variety of different kinds of interactive multimedia system, such as real-world task support (Alty and Bergan, 1992), well-defined simulations (Riddle, 1990) and adventure games (such as *Myst* from Broderbund Software Inc and Cyan Inc, 500 Redwood Blvd, Novato, Ca.), and we drew on aspects of all of these, while also bearing in mind our own design considerations for the system. The final environment is not like any other educational system we have found.

Knowing that our students are very focused and simply would avoid using the system if it took too long to learn, we wanted the environment to be quick and 'intuitive' to learn. We went through several cycles of evaluation and change using story-boards. In particular the general screen layout evolved through various versions, and the initial presentation of the characters and the offices was much more detailed than the final, simple 'cartoon' style of characters. In addition, the course is aimed at professionals involved in software engineering and so our students are likely to be sophisticated computer users. This creates an interesting set of design trade-offs.

We wanted to stay within the software house metaphor as far as was practical, but we were also willing to compromise if appropriate. For example, we initially required the user to 'go into a lift' in order to move between project rooms, but found that this became frustrating and we introduced 'quick keys' as an alternative.

The overall screen layout is shown in Figure 1. The top two thirds is a 'window' into an OSS project through which students interact with other OSS staff and access project resources. The other third is a control panel with the 'quick keys', a button to enable the sound or disable it and display summary text in a speech bubble, an audio/text progress display, and a workbook. The buttons on the left depend on the project room, and transport the user to meetings outside OSS offices, for example to interview the client. The workbook is an active document which users take everywhere with them. This can be used to take notes, receive feedback from the project manager, ask questions in interviews, complete evaluation forms and answer questions.

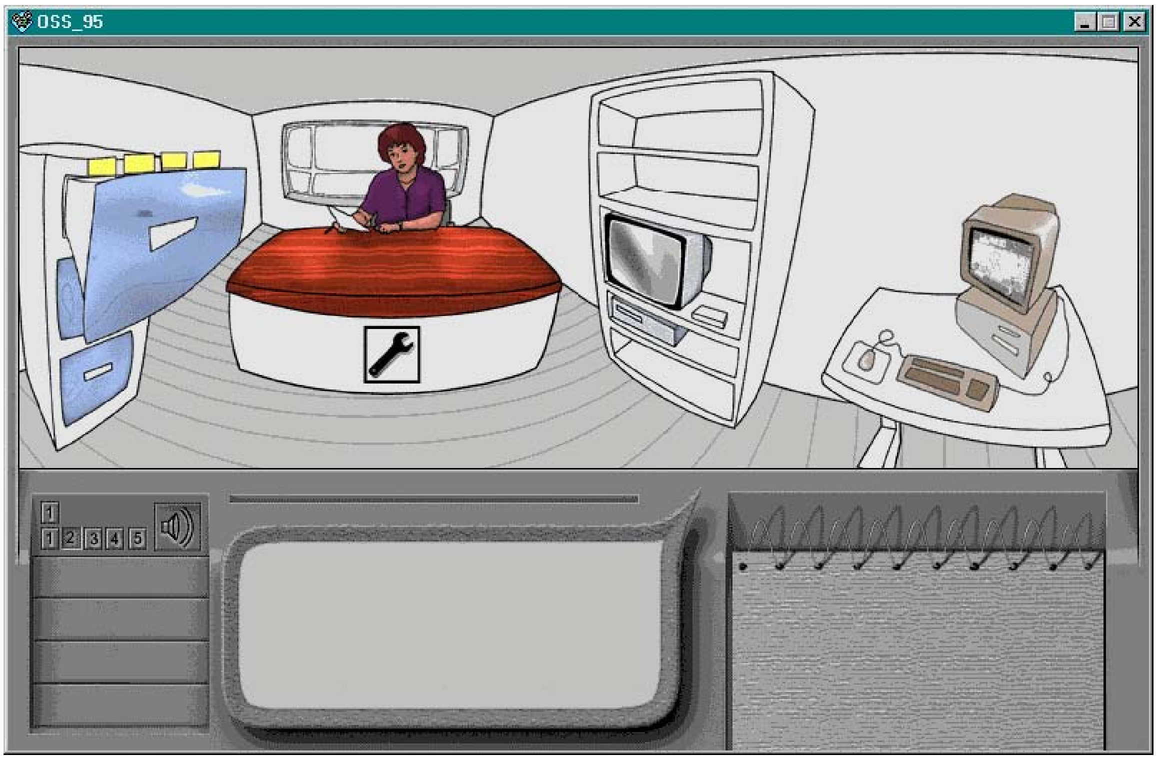


Figure 1 Screen layout (showing the Production Cell office)

Each project within OSS has its own floor in the building, reached via a lift, or using the ‘quick keys’ in the control panel. The ground floor contains a reception, and the top floor contains a library. On each floor, the perspective of the user is of someone standing in the middle of the room. Moving the cursor towards the edge of the screen causes the view of the office to rotate. Each project floor has basically the same layout, and contains the project manager (who provides guidance and feedback), and all the resources necessary to complete the tasks associated with the projects, e.g. books and papers, prototype systems, simulations, access to meetings and so on. Which resources are included depends on the project.

The main screen colour is grey, and hotspots are indicated by colour, with cursor hinting as the cursor moves over them. A pointing hand shows that the item will respond to being clicked on, an open hand shows that it can be picked up and a closed fist shows that it has been picked up and can be dragged. The rest of the time, the cursor shows a drum icon which rotates at the edges of the screen as the view rotates.

The interaction possible matches real-world expectations. Books can be taken from the shelf, placed on the desk, and opened for reading; files can be taken from the filing cabinet and read; the video player can play any video lying beside it, and the computer can be used to run software, such as CASE tools and prototypes. In addition, all the documents can be printed.

In principle things can be done in any order, but we provide guidance through a printed resource guide as well as direction from the virtual project manager. On entering the multimedia, the student is in the reception of OSS which contains a noticeboard with an organisation chart of OSS, the receptionist Patrick, and the lift (see Figure 2). Students can look around reception, and ‘talk to’ Patrick who welcomes them to OSS. They can go up to any of the project rooms via the lift or quick keys.

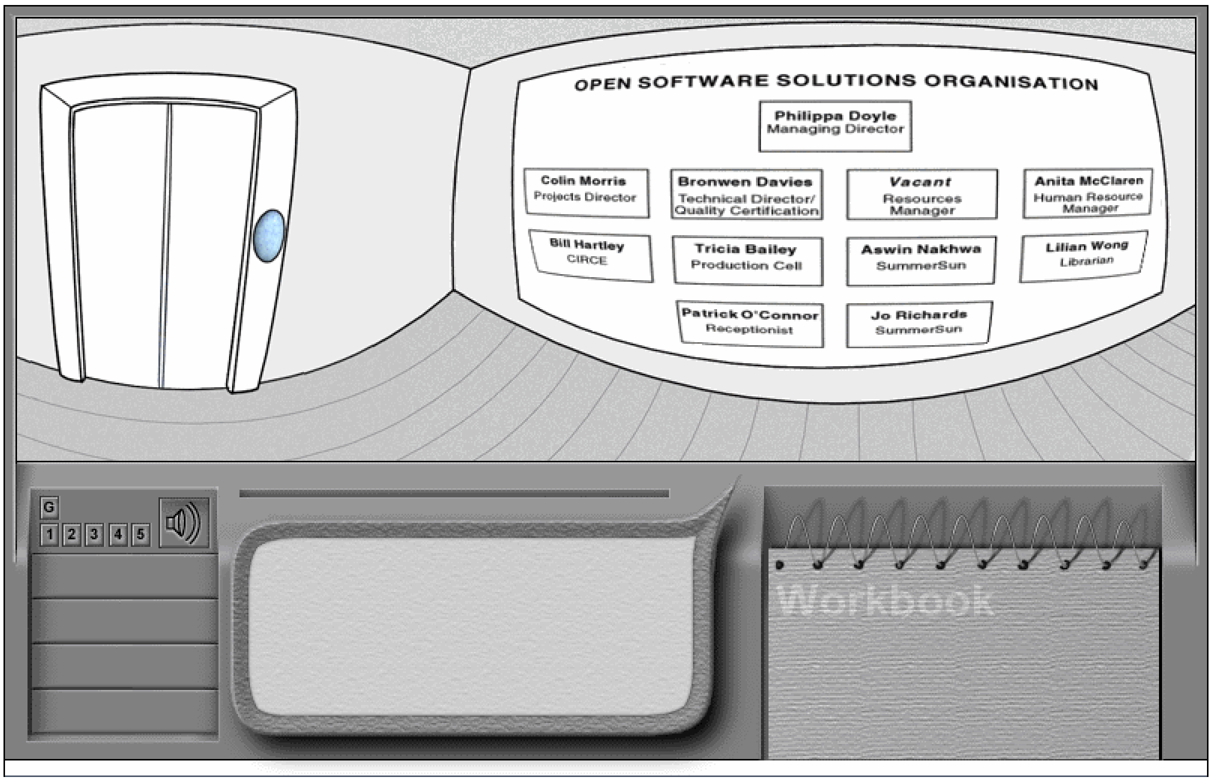

OPEN SOFTWARE SOLUTIONS ORGANISATION
Philippa Doyle Managing Director
Colin Morris Projects Director
Bronwen Davies Technical Director/ Quality Certification
Vacant Resources Manager
Anita McClaren Human Resource Manager
Bill Hartley CIRCE
Tricia Bailey Production Cell
Aswin Nakhwa SummerSun
Lilian Wong Librarian
Patrick O'Connor Receptionist
Jo Richards SummerSun
G
1 2 3 4 5
Workbook

Figure 2 Part of the reception showing the lift door and the OSS organisational chart

Each project manager provides further guidance through pop-up menus and explanations of the tasks and how to achieve them, and offers task feedback.

4. OSS Projects and Tasks

Each project has its own set of resources for the student to use in order to undertake a series of tasks which the project manager will give the student. Each project manager provides instructions and background for the student tasks and feedback once the task has been completed. The workbook has also been pre-loaded with a variety of forms and other resources to support the tasks. In addition the company has a library of materials (see Figure 3) which are more general software engineering resources. OSS uses the two CASE (Computer-Aided Software Engineering) tools which are used in the standard teaching materials. These are Select Yourdon (to support data flow diagramming and structured design), and Select Enterprise (to support the object-oriented lifecycle using UML diagramming notation).

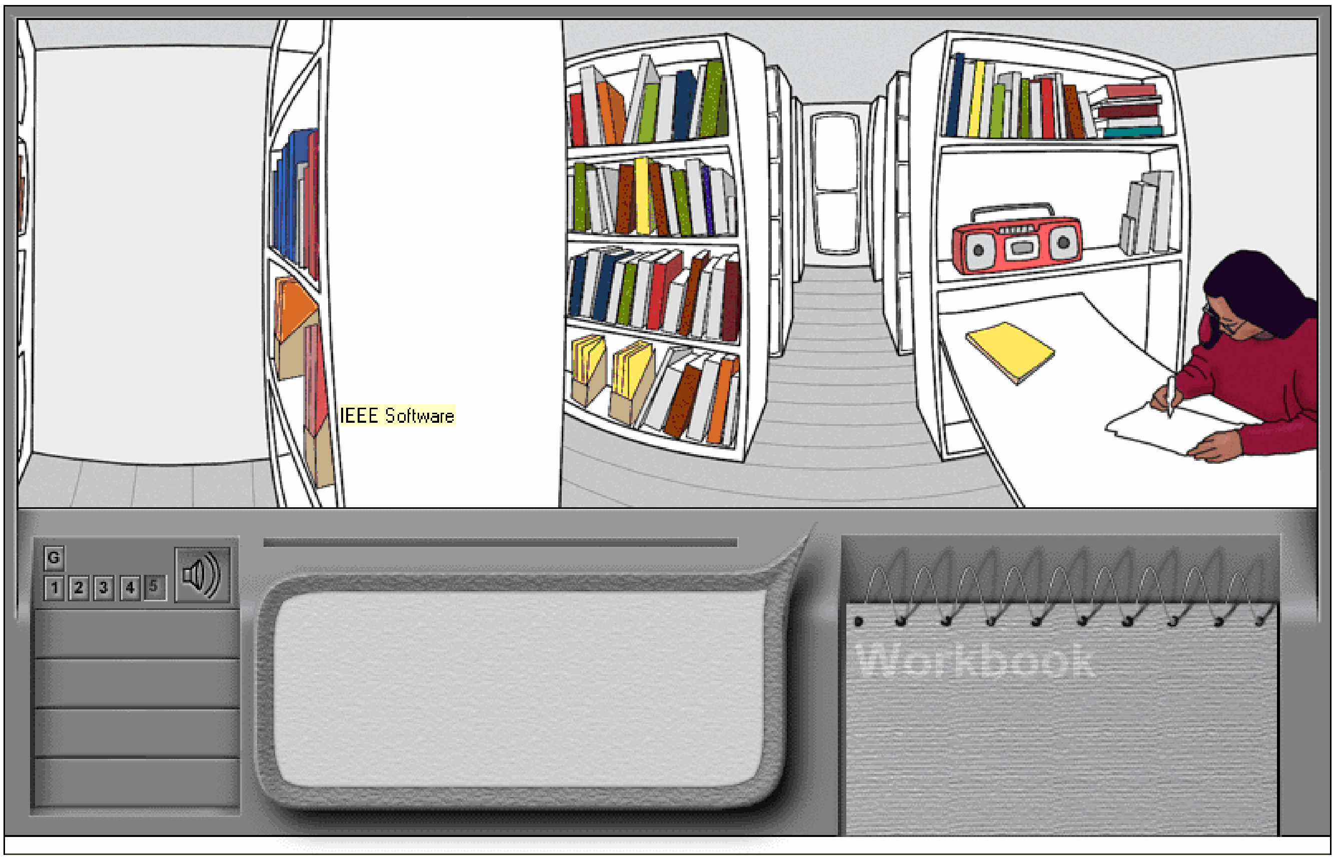


Figure 3 The library on the fifth floor contains several books, journals and manuals related to the course

In this section we describe the student tasks in more detail.

4.1 CIRCE (15 hours of study) – floor one

This project involves three tasks:

1 Familiarity: a chance to get some general background on the project;

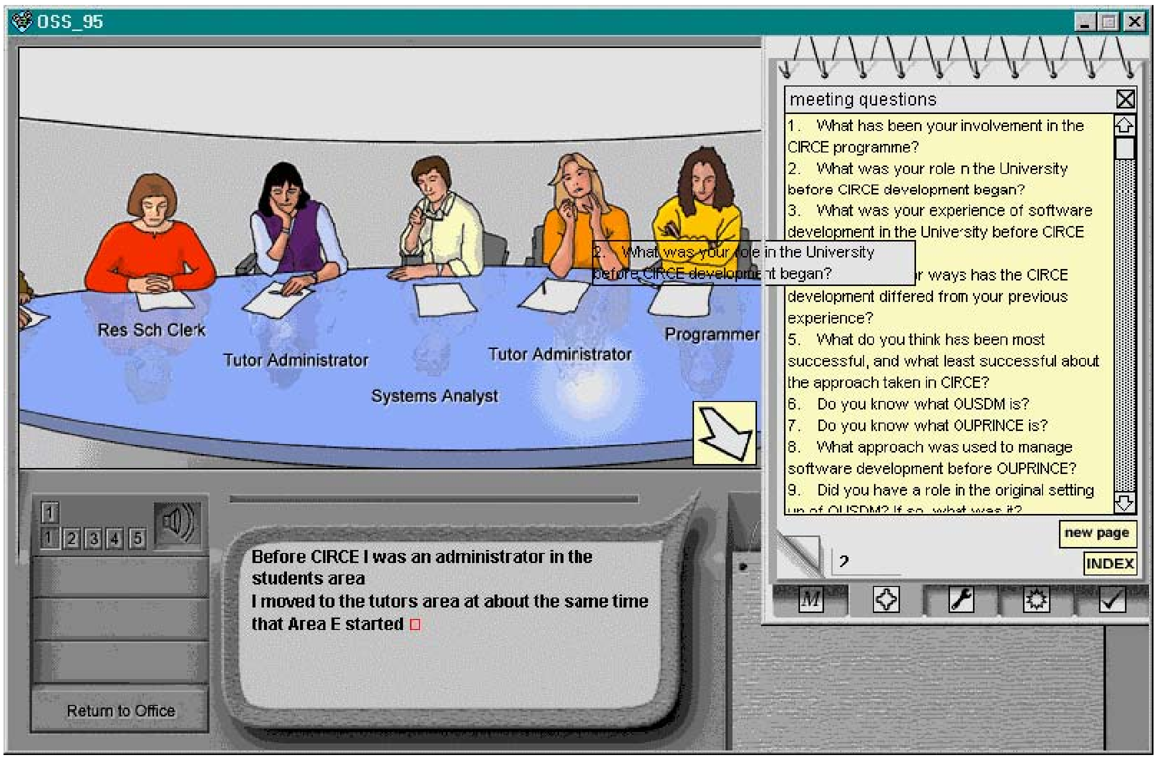
2 Method: an evaluation of the OUPRINCE and OUSDM methods developed for CIRCE;

3 Interface design: an evaluation of the interface design standards and their use in screen design for CIRCE.

The first task requires students to read a general description of CIRCE (located in the filing cabinet), and to look at a slide presentation which sits next to the projector. These two items provide an introduction to the project as a whole in order to give the context for the method evaluation and interface design tasks, which are the main student activities for this project.

The second task requires the student to compare a set of criteria against the views of one of the project sub-teams from The Open University. The criteria were originally developed to guide the development of the new method, and so it is important to check that the final method (OUPRINCE and OUSDM) do indeed meet the criteria. Bill, the project manager, has broken the task down into four sub-tasks. In the first sub-task, students gain familiarity with the methods and put together a set of questions to ask the project team members. To achieve this, the filing cabinet contains a list of the team members and their roles. This team will include users, business analysts, programmers and managers, and each will have their own perspective on the method and its use. The course materials emphasise that it would not be appropriate to ask the same questions of all the participants because of their different roles.

The next two sub-tasks require the student to interview a set of managers and then a project team and complete an evaluation form which compares the interviewees' views with the original method criteria. Figure 4 shows the screen for the interview with the project team. The notebook contains 35 questions which can be asked by dragging the question to an individual, and they will reply either through an audio recording or through the speech bubble in the middle of the control panel. These questions and answers are those actually used in the CIRCE project on which the case study is based. In all, there are 12 people who could be asked each question, but students need to exercise their judgement about who to ask which question if they are not to waste much time trying to ask all questions of all people. In the final sub-task, students complete an evaluation form in the workbook, based on the findings from the interviews.



2. What was your role in the University before CIRCE development began?

1. What has been your involvement in the CIRCE programme?
2. What was your role n the University before CIRCE development began?
3. What was your experience of software development in the University before CIRCE
r ways has the CIRCE development differed from your previous experience?
5. What do you think has been most successful, and what least successful about the approach taken in CIRCE?
6. Do you know what OUSDM is?
7. Do you know what OUPRINCE is?
8. What approach was used to manage software development before OUPRINCE?
9. Did you have a role in the original setting up of OUSDM? If so, what was it?

Before CIRCE I was an administrator in the students area
I moved to the tutors area at about the same time that Area E started

Figure 4 Meeting with a CIRCE project team, showing the workbook being used for interviewing

The third task requires the student to evaluate the interface design of a new prototype system (provided on the computer in the CIRCE office) with the design of the old system (illustrated by a video of current use (see Figure 5), and screen dumps of the existing system). The filing cabinet contains the interface design guidelines and an evaluation form to be completed by the students.

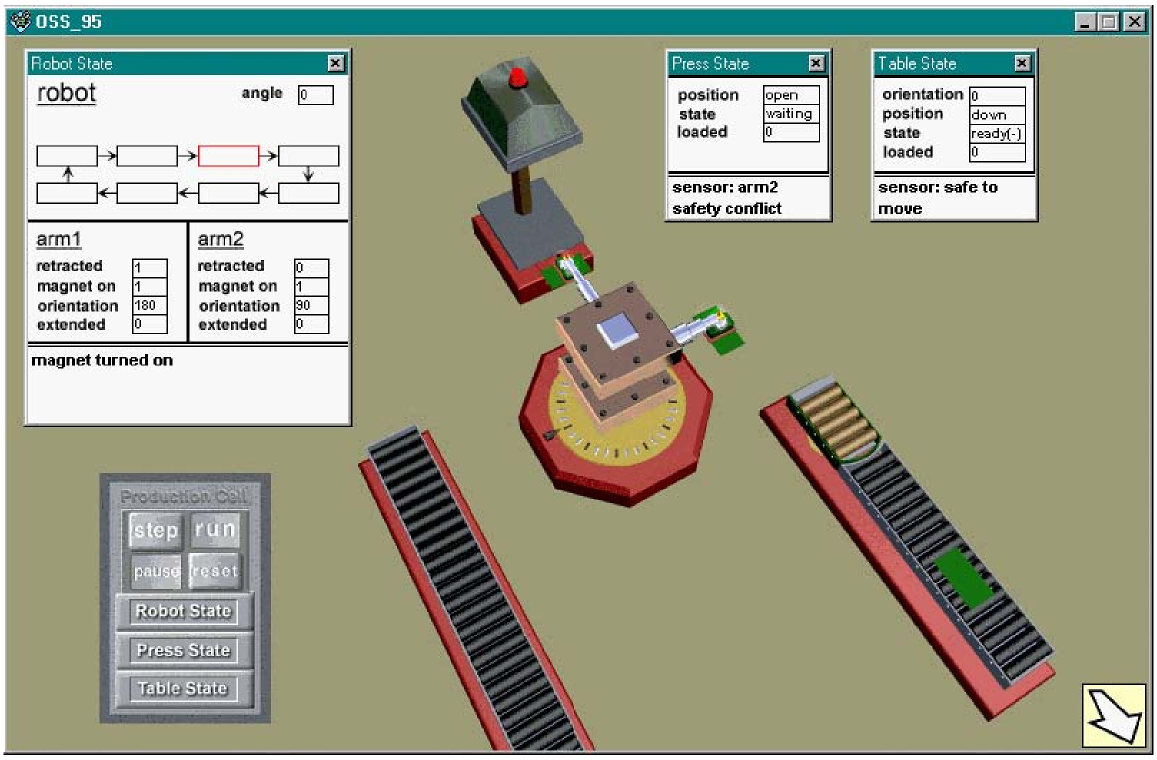
Figure 5. A screen shot from the video illustrating the use of the existing system

4.2 Production Cell (15 hours of study) – floor two

OSS is using this project to train staff in object-oriented techniques and formal methods. The Production Cell is designed to develop students' problem understanding by experimenting with a simulation, and then modelling the observed behaviour.

There are seven tasks here: familiarity, use cases, events, statecharts, operations, safety properties and Z presentation. The first task introduces the problem to the student using notes from the filing cabinet, a video of the production cell in the factory, and a simulation of the production cell and how it works (see Figure 6). The simulation is a key resource in this project and is used throughout all the tasks to investigate the production cell's behaviour, and hence to inform the modelling which students are asked to undertake. Use cases, events, statecharts and operations are all aspects of UML modelling which students are taught through the standard teaching materials, and the project guides them through these steps. The events, operations, safety properties and Z presentation tasks are supported by a slide presentation, each of which is sitting by the side of the slide projector. Students also can use the Select Enterprise CASE tool to draw the UML diagrams created to model the system and its behaviour. There are several other resources used in the tasks including diagrams on the noticeboard, sections of the requirements specification which are copied into the workbook for easy reference while running the simulation, and other documents.

The safety properties task requires students to analyse the model produced by the earlier steps and decide if the behaviour modelled is safe. The Z presentation task asks students to produce some simple elements of a Z specification for the production cell, based on the analysis completed in previous tasks. This project applies directly the modelling, notations and requirements analysis skills taught through the standard teaching materials.



robot angle 0

arm1	
retracted	1
magnet on	1
orientation	180
extended	0

arm2	
retracted	0
magnet on	1
orientation	90
extended	0

magnet turned on

position	open
state	waiting
loaded	0

sensor: arm2 safety conflict

orientation	0
position	down
state	ready(-)
loaded	0

sensor: safe to move

Figure 6 The Production Cell simulation

4.3 SummerSun (30 hours of study) – floor three

The SummerSun project shows the student how various techniques may be used through a development lifecycle – from requirements elicitation to test planning. It is also the only OSS project with a staff member other than the project manager (i.e. Jo Richards the programmer). Through interaction with Jo and the project manager the student gets different opinions about the client and the project, illustrating one of the human aspects of software development.

The project has six tasks: joining the project; requirements elicitation, entity-relationship analysis, reconciliation and review, test planning and leaving the project.

The first task involves reading background information about the client and the project, and also talking with Aswin. Requirements elicitation is based around the memo from the client and identifying functional and non-functional requirements, process constraints and goals. The memo is in the workbook as well if the student wants to annotate the memo in here. Alternatively, the documents can be printed out. Jo and Aswin have produced their own list from the memo and when the student has finished, Aswin will compare the two sets of results and explain to the student how they arrived at their list. Having compared these two lists, the next sub-task is to visit the client and interview members of the SummerSun travel agency, in order to compile a final list of requirements. This final list is compared with Aswin's set of requirements, before moving on to the next task, which starts the modelling phase.

Task 2 utilises the Select Yourdon CASE tool to produce an entity-relationship (ER) model of the system's database. Aswin supports the student through this task by dividing it up into smaller chunks focusing on only a small portion of the system initially, and providing his own models for comparison and reflection. First, students are asked to identify a list of candidate entities, second to finalise the list of entities (after comparison and explanation), third to identify relationships between entities and produce an ER diagram in Select Yourdon. In the fourth sub-task the student compares Aswin's ER diagram and their own, and in the fifth the student is asked to complete the entire ER diagram for the system.

In parallel to this activity, Jo the programmer has been developing a prototype system. Task 3 in this project involves reconciling the ER analysis and the prototype to see if they have uncovered different aspects of the system, and considering how best to integrate the two perspectives. Jo's prototype can be run on the computer in OSS's SummerSun office.

The basis of this reconciliation and review is a 'CRUD' matrix and an 'Outline Traversal' matrix, taught through the standard teaching materials. The CRUD matrix compares each entity from an ER diagram and documents where the entity is Created, Read, Updated and Deleted. It is a simple mechanism to ensure that the entities are all treated appropriately – e.g. that no entity is updated or deleted without being created; if it is created and deleted but never read, why is it in the system? The CRUD matrix checks entities and their definition, while the Outline Traversal matrix is used to investigate relationships between entities. Aswin introduces this matrix and explains how to complete it.

In the fifth task, test planning, students are asked to prepare test cases for the main function of the system, i.e. booking a holiday. Again, Aswin provides structured help by breaking the task down into several sub-tasks and providing his own solutions for comparison.

The final task is designed to summarise what the student has done in this case study and is delivered in the form of a presentation from Aswin which reviews the lifecycle used in OSS (see Figure 7) and the outcomes generated.

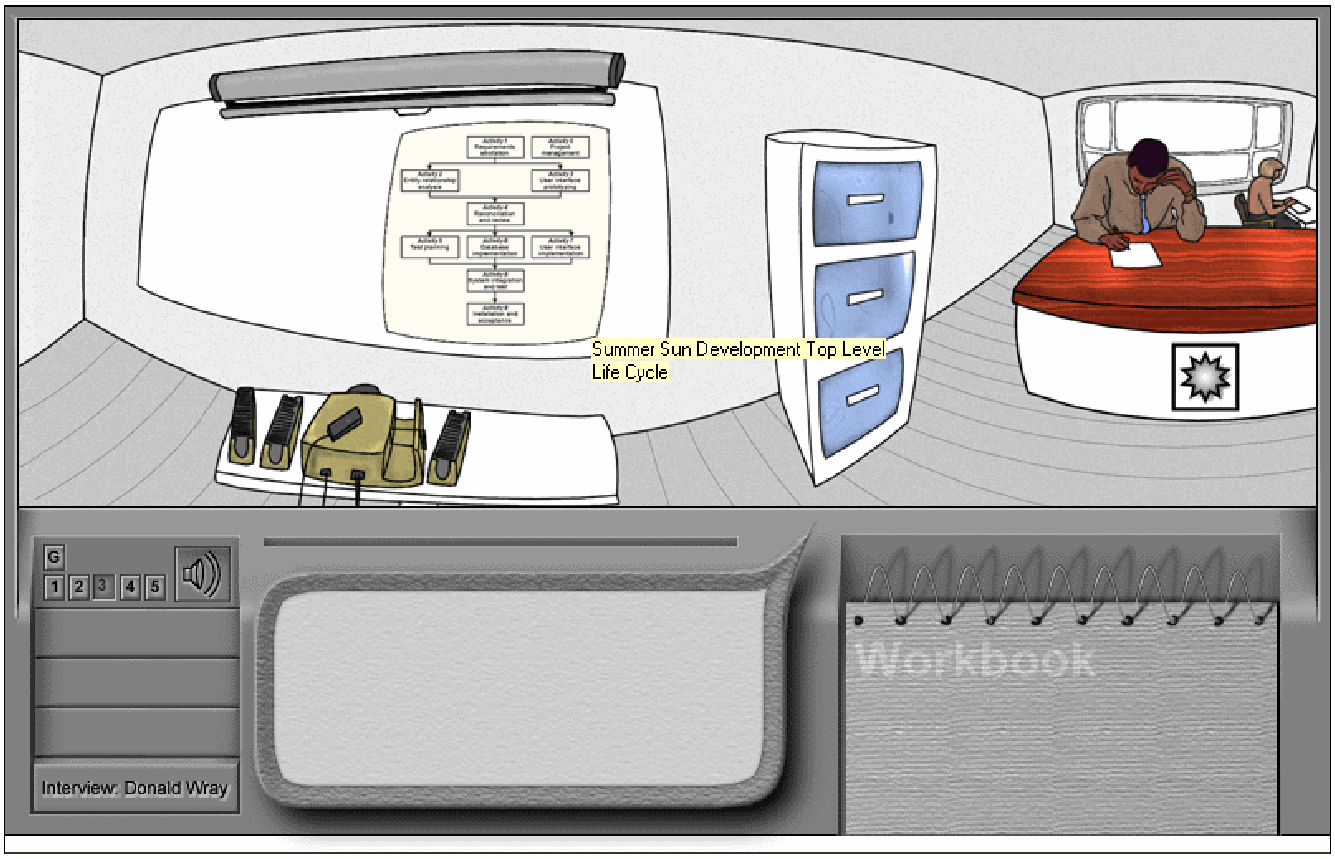


Figure 7 The Summer Sun office showing Aswin and Jo, the noticeboard and the slide projector

4.4 Quality Certification (10 hours of study) – floor four

This project is focused on improving OSS's internal processes and getting certification. This will allow the company to bid for projects with government organisations and to work with safety-related and safety-critical systems. This work is related to the Production Cell project where staff are being trained in safety-related matters. OSS has an existing draft quality manual, but it is in need of review and updating, which is the basis of the student's tasks. There are four tasks in this project. Initially, students are asked to list the sections they would expect to find in a quality manual for this company (based on a list provided in the standard teaching materials) and then compare the existing manual with this list. As with previous projects, the feedback for this task is based around a solution that the project manager (in this case Bronwen Davies) has already produced, and her explanation for it.

The second task in this project is a more formal evaluation of the draft quality manual, based on a list of procedures and objectives which Bronwen believes should be covered. The student is asked to provide a cross-referenced list between the sections of the manual and Bronwen's list of procedures and objectives. A blank cross-reference table is provided in the filing cabinet, and a completed one is also available once the student has completed the task. The student is asked to identify gaps in the quality manual. In the fourth task, the student is asked to draft the missing procedures, supported by Bronwen's comments and other documents in the filing cabinet. In particular, the testing procedure is missing, and Bronwen supplies an outline procedure for the student to complete.

Finally, some measurement detail (omitted in the outline used in task 3) needs to be added in order to complete the control aspects of the procedure.

5. User evaluation

M880 was first presented in November 1997. The OSS environment underwent testing and evaluation before the start of the course, and then after the first presentation we issued students with the MUMMS (Measuring the Usability of Multi-Media Software) questionnaire (http://www.ucc.ie/hfrg/questionnaires/mumms/), based on SUMI (Software Usability Measurement Inventory) (Porteous et al, 1993) to assess the usability of the environment. Unfortunately there were some technical issues with the environment for this first presentation, specifically that the software did not always install correctly, and software conflicts arose regarding audio and video files. These were corrected by the second presentation, but these difficulties may have affected the usability evaluation presented below. A second evaluation carried out via a survey instrument after the second presentation focused on pedagogical as well as usability issues, and is reported in Section 5.2.

5.1 The MUMMS (usability) analysis

Summary statistics from the MUMMS analysis of the data from the first course presentation is given in Table 1. Note that this is based on 31 responses. Figure 8 illustrates these results graphically. Below this table and figure is an extract from the commentary accompanying the MUMMS analysis.

Table 1 Summary statistics from the MUMMS usability analysis

	Attractiveness	Control	Efficiency	Helpfulness	Learnability	Excitement
Average	48.72	56.93	50.58	53.04	54.63	55.70
Median	46.83	59.33	51.00	55.17	59.33	55.17
Standard Deviation	18.22	16.73	17.60	17.14	19.00	14.90

SE Mean	3.52	3.52	3.52	3.52	3.52	3.52
Max	80.17	80.17	80.17	80.17	84.33	84.33
Min	26.00	21.83	21.83	26.00	17.67	17.67
N	31.00	31.00	31.00	31.00	31.00	31.00

"Comparing the Average (Mean) and the Median: for Attractiveness there appear to be some users who are giving much higher ratings than the rest of the group (elevated Mean); for the other scales there seem to be groups who give much lower ratings (depressed Mean).

"Standard deviations are however quite large, indicating a wide spread of opinion (they should be around the 10.00 level and they are all much higher than that). This is also amplified by considering the maxima and minima: essentially, all shades of opinion are demonstrated for each of the scales."

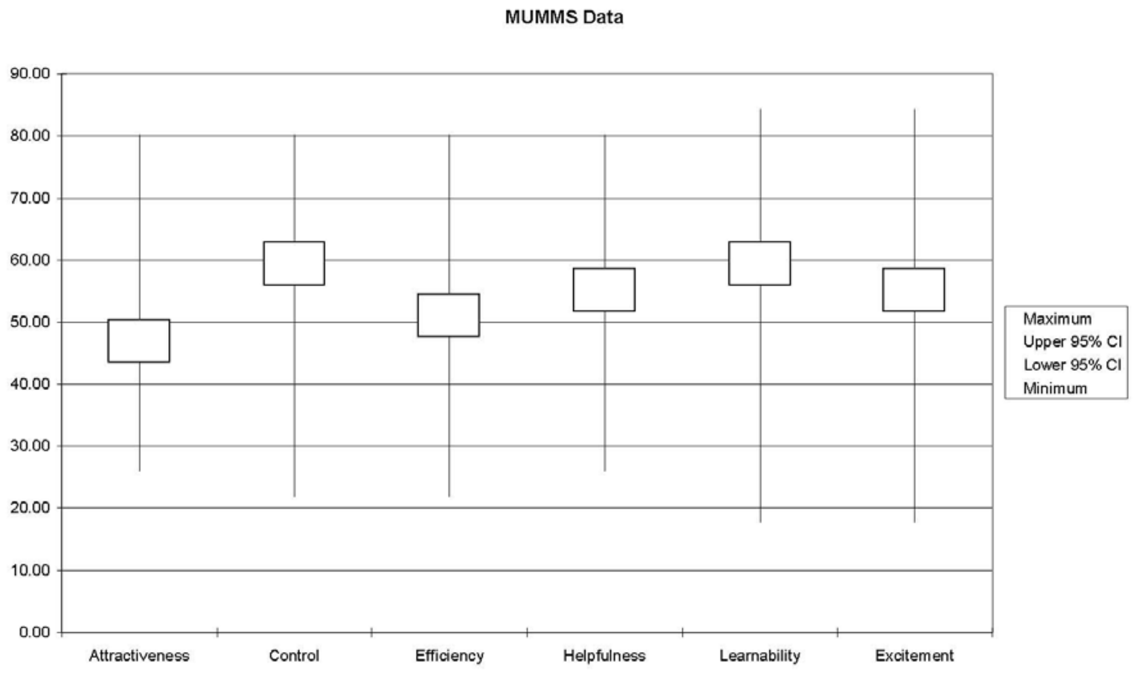


Figure 8 Summary statistics from the MUMMS usability analysis in graph form

"The system rates low on Attractiveness, below the expected population mean of 50, although since the 95% confidence interval just touches the 50 line this is not as bad as it seems.

"The strongest suits of the software are its Control and Learnability: users feel they are in control of the interaction, and they feel the application helps them overcome operational problems with it."

The questionnaire also asked for comments about positive and negative features, and here are some examples to illustrate specific successes and specific problems.

Positive features	**Negative features**
well designed + structured, great fun easy to use	adds nothing new, simply a distraction, time consuming

very friendly, makes learning less boring	fish eye view of environment, flat colours
easy navigation, clear	lack of guidance and poor explanations, bad configuration instructions
based on real people	poor documentation, confusing
comprehensive information, good graphics + sound	accent of voices
provides a framework for the course, good simulation	long winded, no short cuts, lack of assistance and explanations
welcome change of media	

As pointed out in the MUMMS commentary, there is no clear finding relating to the overall usability of the environment. Students differ considerably in their reaction to the system, and this is underlined by the mixture of positive and negative comments. It should be noted that some of the negative comments regarding documentation and lack of guidance may be referring to the technical problems the first presentation faced (as described above). However, the analysis also highlights that the learnability and control are the software’s ‘best suits’, and making the software easy to learn and use was one of our aims.

5.2 The second course evaluation

At the end of the second presentation (November 1998) we surveyed our students again using The Open University’s own standard questionnaire which is issued to all students who have studied a new course. This questionnaire covers a variety of issues from tutor support and timely delivery of materials, through to how well the course fulfilled its learning outcomes and how enjoyable it was. It also includes a set of questions specific to the individual course, and chances to provide free comment on any aspect of the course. Out of 354 registered students, 126 questionnaires (36%) were returned.

One question asked about the overall usability of the software. 69% of respondents claimed that the software was ‘fairly’ or ‘very’ easy to use, reflecting the MUMMS analysis regarding control and learnability.

The results for ‘How much value was OSS’ were disappointing with 23% saying ‘Not at all’, 46% saying ‘not very’ and only 7% saying ‘very’.

When asked if they would prefer the speech in text bubbles or audio, 50% expressed ‘no preference’, 31% preferred text in bubbles and 18% preferred audio.

One of the design goals of the environment had been to make the environment as realistic as possible. However, 37% of respondents claimed that the realism was ‘not at all’ important, and 63% said that it was ‘not very’ important. This means that none of the students saw realism as being important.

Only 19 students out 115 free text replies chose to comment on the multimedia directly. Negative reactions were: ‘Thought OSS was drivel’, ‘OSS was appalling’ and ‘At post graduate level I expect to be guided in my study- not given children's games to play with’. Positive reactions were: ‘Made parts of the course entertaining’ and ‘I think that the provision of realistic case studies via the CD material is an excellent feature of the course and should be continually developed’.

5.3 Evaluation Discussion

The MUMMS analysis highlights the strengths of the system as being in its control and learnability. This is echoed in the subsequent evaluation with users claiming that the environment was easy to use, and specific comments such as being easy to navigate and very friendly.

Given that the case studies covered half the course study time, it is surprising that students regarded the value of OSS as being so low, but it is possible that the case study material itself was helpful, but the packaging of it in OSS was not so attractive. From the second evaluation comments, it is clear that feelings ran quite high among some students who felt insulted at the style of the environment (‘children’s toys’), while some regarded it as fun and a welcome change of media.

Despite our own concern that the environment should be realistic, none of the students who answered the survey regarded this as important. Unfortunately no further comment about this was provided and so we are unable to offer any further analysis of this aspect. Our speculation is that the students had not expected the case studies to reflect (or relate to) their own experiences on real projects and were content to keep their everyday working lives separate from the course. In other CCI courses, we have found that the ability to link assessment and course learning with real experience strengthens and deepens the student’s understanding, and so this is something to be encouraged in M880.

During the development, we spent considerable time discussing the provision of speech in audio and text forms in order to provide students with the flexibility to run the environment in a quiet environment, or on the train, and to account for various disabilities. It turns out, however, that most of the students did not regard this as a particular issue.

6. Conclusions

Our initial question was whether we could use interactive multimedia to give real project experience that would enhance our students’ learning. The evidence we have gathered indicates that, for some students, the environment was stimulating and useful, while for others it was little more than a game. Surprisingly, students we surveyed did not feel that it was important to make the environment realistic.

From our own perspective, the OSS environment we produced fulfilled several of our design aims, including a range of projects with substantial tasks attached to them, a variety of roles and characters, a number of techniques and principles used in combination on complex projects, a sense of some human aspects of software development, and in an environment that was easy to use. From the user evaluations it seems that our users agree that the environment is easy to use, but there are few comments regarding the efficacy of the case studies themselves, and the OSS environment has received a mixture of responses, across a wide spectrum from very negative to positive.

One of our conclusions is that the case studies need to be better integrated with the standard teaching materials. Although the relationship was described in the course guide and the study calendar directs students to study certain projects at certain times, there is a sense that students do not understand the relevance of the case studies to the rest of the course. We therefore decided to integrate the case studies into the course assignments more closely, thus strongly encouraging the students to engage with the material.

A second conclusion is that the environment itself was trying to include too much material. Although one of our aims was to communicate the complexity of software development, our students clearly did not expect this complexity. For example, one of our students printed off all the material in the CIRCE office (several hundred pages of manuals and standards) and then rang up their tutor and said ‘what do I do now?’. Students seem to be uncomfortable with an environment where they have a lot of information, and lots of freedom, but are expected to exercise their own judgment and to take guidance from an electronic avatar figure. In a mixed-mode environment where face-to-face tuition and online learning are combined, the students could be guided through the case study material in a more structured fashion, and we feel that this may alleviate this problem. Although the project managers within the environment performed this role, it appears to have been unsuccessful.

Finally, trying to capture human aspects of software development in this environment was difficult and although we managed to reflect some aspects, its paucity seems to have resulted in some students dismissing rather than appreciating the problems.

In any future development of this environment we would seek to present tasks in smaller chunks in a more controlled way, thus simplifying the environment as a whole.

Acknowledgements

The environment described in this paper was produced by the BBC's OU Production Centre. All members of the M880 course team contributed to the success of this project.

References

Alty, J.L. and Bergan, M. (1992) "The design of multimedia interfaces for process control" in *Proceedings of 5th IFIP, IFAC, IFORS, IEA Conference on Man-Machine Systems*, 249− 255.

Boyle, T. (1997) *Design for Multimedia Learning*, Prentice Hall Europe.

Bransford, J.D., Sherwood, R.D., Hasselbring, T.S., Kinzer, C.K. and Williams, S.M. (1990) "Anchored instruction: Why we need it and how technology can help", in D.Nix and R. Sprio (editors) *Cognition, education and multimedia*. Hillsdale, NJ: Erlbaum Associates pp 115-141.

ED-Media 99 (1999) *Proceedings of ED-MEDIA 99*, Association for the Advancement of Computing in Education.

Laurillard, D. (1996) Keynote speech at the 1st Conference on Integrating Technology into Computer Science Education (ITiCSE), Barcelona, Spain, 2-6 June 1996.

M880 Software Engineering (1998), The Open University, UK. Details available from the authors' address.

McLellan, H. (1995) *Situated Learning Perspective*. Englewood Cliffs, NJ: Educational Technology Publications.

Porteous, M., Jurek, K. and Corbett, M. (1993) *SUMI User Handbook*. Human Factors Research Group, University of Cork, Ireland.

Pressman, R. (1994) *Software Engineering: a practitioner's approach*, 3rd edition, European Adaptation, McGraw-Hill.

Riddle, D. (1990) EcoDisk CD-ROM, *CTISS File*, No 10, September 1990.