Teaching Systems Analysis and Design Using Multimedia and Patterns

Jacob L. Cybulski and *Tanya Linden*

j.cybulski@dis.unimelb.edu.au
http://www.dis.unimelb.edu.au/staff/jacob
ph: +61 3 9344 9244
fax: +61 3 9349 4596

t.linden@dis.unimelb.edu.au
http://www.dis.unimelb.edu.au/staff/tanya
ph: +61 3 9344 9250
fax: +61 3 9349 4596

Department of Information Systems
University of Melbourne
Parkville, Vic 3052, Australia

Abstract

Research in educational multimedia shows many benefits of learning with the use of multimedia environments. To take advantage of these benefits we developed Multimedia Assisted Teaching Environment (MATE), which is currently trialled in supporting our teaching of IT subjects. MATE's main strength is its ability to facilitate reuse of multimedia components while developing teaching material. In this paper, however, we describe MATE from the perspective of a student learning Systems Analysis and Design via projects and case studies. We illustrate the workings of the MATE tool with an example that shows our approach to teaching design skills with reusable design patterns.

1. Introduction

Systems analysis and design (SA&D) is an established area of teaching in any information technology (IT) course. Our experience indicates that compared with teaching some other IT fields, e.g. programming, SA&D is notably more difficult. One of the reasons for this difficulty is that to analyse and design a computer system effectively, one requires not only excellent technical skills but also a lot of knowledge that goes beyond the technical boundaries of IT, e.g. knowledge of business, management, politics or psychology. Practicing systems analysts gain such knowledge by gaining hands-on experience while dealing with problems in systems development, by interacting with users, managers and developers, and by sharing experience with other more accomplished analysts. Needless to say, beginners have only a fraction of expert's knowledge, they also lack any significant practical experience that could give them some insight into the real issues in systems analysis and design.

A common approach to teaching systems analysis and design is for teachers to facilitate the transfer of experience from experts to students by means of real life examples and case studies.[1] Such simulated situations allow exposing students to various design approaches and possible problem solutions. One of the difficulties with the case study approach is that there are usually

[1] This assumes that teachers have experience that could be transferred. One of the main problems in teaching SA&D is a great shortage of support staff, such as tutors and lab supervisors, who have suitable knowledge and practical experience.

several different ways of arriving at a correct solution to the problem and there exist many valid techniques applicable. Tutors are commonly (and intensively) involved helping students identifying examples of good and bad analysis and design techniques. Such an approach is not optimal, as students need to develop their own abilities to judge the value of each possible approach and to select the best practice for the discussed case. Another, more effective, way of learning analysis and design skills is to solve problems in teams, where students could be involved in debating differing approaches to solving the problem, negotiating the common strategy and then collaborating on the actual problem solution. SA&D is one of the subjects that cannot be easily learnt by self-directed study of textbooks, nor by conducting small-scale exercises. Learners in SA&D require to be involved in the process of analysis and design of a large system, they need to work in teams, and they must have nearly unimpeded access to advisors and tutors.

As many other teachers of SA&D, we currently teach the subject in a traditional way, by means of lectures, tutorials and laboratory sessions. In lectures, a lecturer communicates substantial amounts of information in a short time. The lecturer is presenting slides and explaining the material while students are listening and taking additional notes, but make limited contribution to the discussion of presented information. Due to large class size (250-300 students) students are assigned passive role in lectures. As a result students absorb only about 30% of the concepts presented in lectures [7, 8]. We are trying to compensate the disadvantages of lectures with collaborative problem solving, guided solution design and discussions that happen during tutorials. In laboratory sessions students put their knowledge and skills into practice, by using CASE tools to model solutions to problems introduced in case studies. Outside scheduled classes, instructors provide additional help to students experiencing difficulties. Time limitations and big numbers of students requiring assistance, however, prevent us from offering high quality assistance. There is much to be said of the shortcomings of this traditional approach to teaching SA&D, especially when large of numbers of students are involved!

To improve the effectiveness of teaching large groups of students, some instructors turn to the use of multimedia. Multimedia educational environments cannot resolve all of the problems experienced by SA&D teachers. In particular, multimedia cannot replace live teamwork nor it is a substitute for working in a real-life organisation. It can, however, be effective in dealing with some educational issues, e.g. students enjoy learning with multimedia, they frequently prefer multimedia material over traditional instruction, many students believe that multimedia improves their learning outcomes [10]. Research in educational multimedia shows many advantages associated with learning via educational multimedia environment, these include organisational, instructional, effectiveness and efficiency aspects [2], e.g.

- Learning appears to take less time when multimedia instruction is used [10].
- Use of interactive multimedia can reduce teaching costs and improve efficiency, it can increase availability and flexibility of teaching resources [2, 3, 10].
- Interactivity appears to have strong positive effect on learning [10]. Interactive multimedia educational environments maximise learners' decision making opportunities [11], which is invaluable in teaching SA&D.
- The learner can control the pace of learning [9, 10], which is an advantage, especially for fast learners.

- Computer-based multimedia environment is capable of presenting information in multi-modal fashion, i.e. by combining auditory, visual and text information. Research shows that students learn more from such multi-modal presentations [1, 10-12].
- Compared with a traditional lecture, computer-based teaching forces instructional designers to better organise and structure educational material. This improved information organisation may be responsible for some of the learning advantages associated with multimedia environments [10].

Having considered these benefits of using interactive multimedia in learning and teaching, we decided to incorporate it in our approach to teaching of SA&D.

2. Our Approach to Teaching SA&D

In the process of systems analysis and design, developers commonly encounter design situations that re-occur from one project to another. Some of these situations relate to the fundamental design principles that call upon well-known design solutions. These problems with explanation and evaluation of solutions can be, and some argue that they should be, recorded in the format of design patterns [4]. We can say that patterns are design principles captured by experts in the field based on their experience. Software development patterns are created, discussed and constantly improved by the pattern community in meetings, workshops and conferences, e.g. ChilliPLoP, EuroPLoP and PloP.

Since patterns arise from successful and unsuccessful practical experiences, they prove to be very helpful in developing high quality solutions to problems found in real life situations, which are frequently the very same problems that we present to students as their project themes and their case studies. One of the major benefits of using design patterns in software development is that they communicate expertise to non-expert developers - which is why we decided to use patterns in our teaching of systems analysis and design. Our teaching experience shows that design patterns can be used as a vehicle to teaching some of the most important techniques in systems analysis and design. The techniques which include analysis of problems, identification of alternative solutions to these problems, and selection of the best problem solving method out of several possible candidates.

To better facilitate the pattern-based learning process, we also developed a multimedia environment capable of guiding students in the effective use of design patterns. Multimedia Assisted Teaching Environment (MATE) supports our students in all stages of their systems analysis and design study - from the preliminary stages of examining previously solved problems to a stage of being confident enough to independently develop project solutions and submitting them for assessment. The tool is highly interactive, hence, it facilitates students' active participation in the learning process. The environment also provides students with access to examples and case studies that could be studied outside scheduled classes. Such examples are used to deepen students' experience and to provide students with help in their project work.

Since our teaching of systems analysis and design focuses mainly on object-oriented methods of systems analysis and design, all our examples are expressed in terms of Unified Modelling Language (UML), which is used for the systematic visualisation, specification, construction, and documentation of software systems and their components [6].

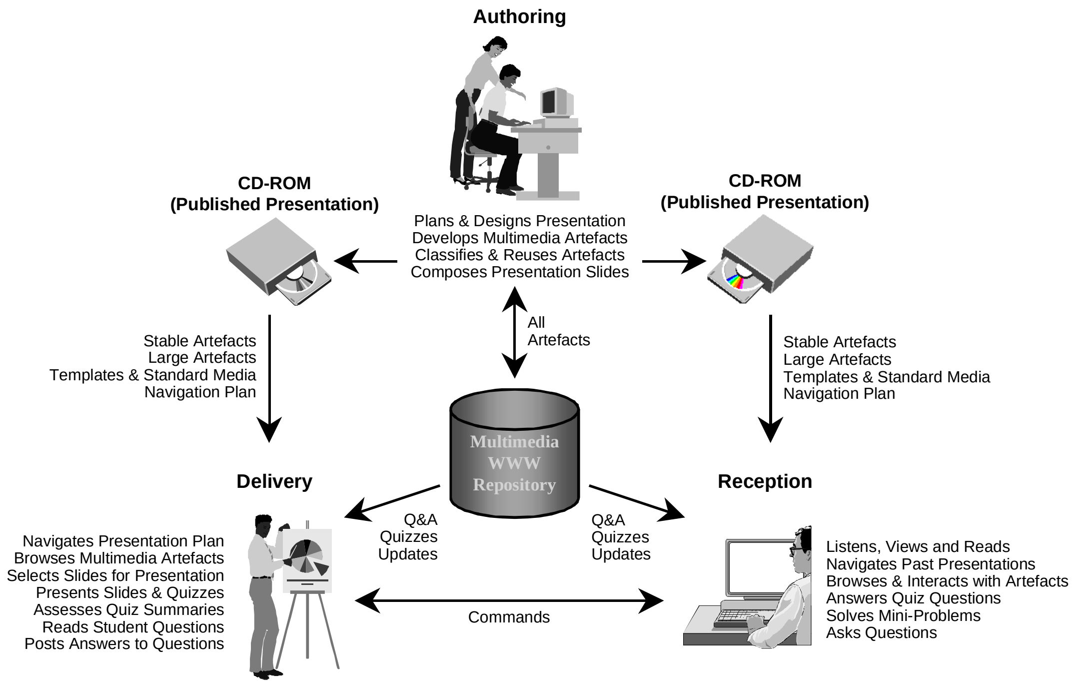

Figure 1 - MATE architecture

3. MATE: Concepts and Architecture

Multimedia-Assisted Teaching Environment (MATE) is an integrated software environment that covers three activities of teaching and learning with the use of multimedia (see Figure 1):

a) **Authoring** includes creation of multimedia teaching components by writing, drawing, composing and reusing those components in a presentation. In the process, authors develop a large database of SA&D case studies, together with problem descriptions, their fully developed solutions, and design patterns that can guide developers in arriving at a solution.

b) **Delivery** of teaching and assessment material by a presenter, who navigates through the presentation, narrates the slides and answers students' questions. In a presentation, a SA&D teacher has an opportunity to illustrate lecture concepts with the material drawn from case studies or the material that is also available for students' self-directed work.

c) **Reception** involves active learning and evaluation of acquired knowledge by students in remote locations. The learner listens to the lecture and views the slides, asks questions, browses through the material independently of the presenter, or undertakes drills, quizzes and other types of assessments.

MATE helps both teachers and students to transfer knowledge and experience in systems analysis and design with the use of multimedia.

4. MATE from Student's Perspective

From a student's perspective, the MATE environment allows active participation in the lecture regardless of the student's location. The participation may take a form of asking questions to the lecturer, providing answers to lecturer's questions, or discussion of concepts and case studies. Students may also choose a self-paced study mode. They can browse through the case studies with fully developed solutions or they can choose to construct solutions themselves with or without guidance from the system. Help is available at any stage of solution development. It may take the form of feedback on student's actions explaining why the step in solution development is correct or incorrect, or the student may be shown a solution to a problem from a related or similar case study. In general, MATE supports a student in the process of:

- **Learning** - by studying solutions to existing case study problems, by creating own solutions with some feedback from the system, by guidance from the system throughout the process of problem-solving.
- **Gaining experience** - by working on case studies and real-life examples, by comparing their solutions to those of the similar problems, and by solving mini-problems and quizzes. In the process students learn from their own mistakes. Having committed mistakes, they can correct their them based on the feedback received from the system and from their instructors.
- **Design** - by creating solutions to case studies. At each point in solution development a student has to make design decisions where several decisions may be correct but they will lead to different design models.
- **Scavenging** - by browsing through the space of existing problem descriptions, problem-solutions, design-patterns and design-rationale.

To illustrate student's work in self-teaching mode while constructing a solution to the problem, we will discuss a mini-case study adopted from the prescribed text "Applying UML and Patterns" by Craig Larman (see Box 1). This example demonstrates how the use of design patterns assists students in designing UML models starting from project requirements to the final set of diagrams modelling the solution to the problem.

Initially students are presented with requirement specifications. After reading and understanding requirements they shown a use case diagram editor (Cf. Figure 2). Students are then expected to select

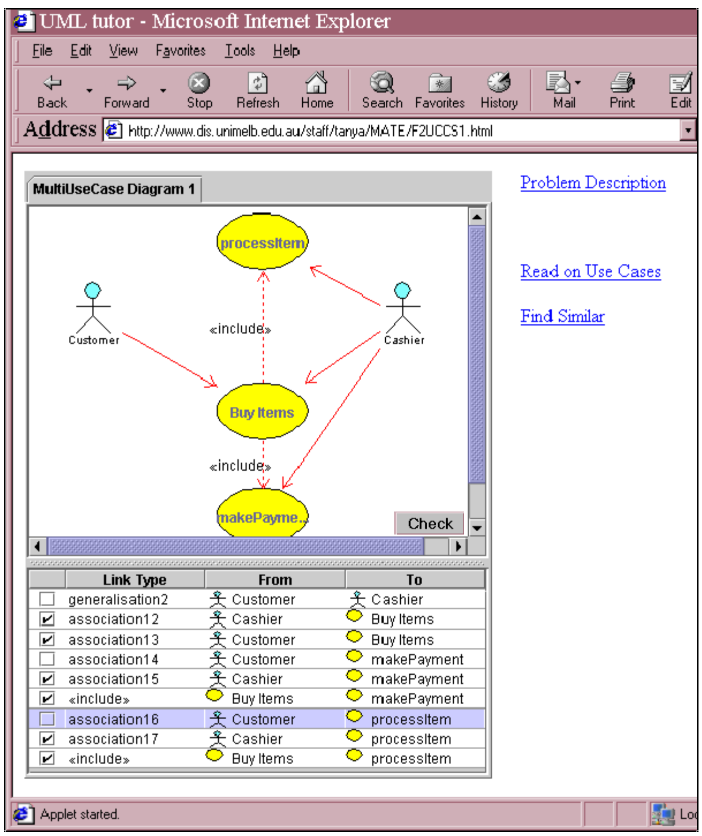

Figure 2 - Use Case Diagram Editor

several valid actor-use-case links from the list of pre-defined candidates. Such links would have been pre-defined links by the author of the case study based on the requirement specifications.

In the example described, a student is expected to correctly select an association between the actor Cashier and the use case Buy Items, association between the actor Customer and the use case Buy Items, and an “include” link from the use case Buy Items to the use case Process Item. The editor produces a diagram reflecting the selection of components.

An application is required to run a point-of-sale terminal (POST). The POST system should be able to process sales and handle payments. When processing a sale the system should accept and record items purchased and their quantities, calculate cost per item and current sale total. At this stage data entry happens manually and only cash payments should be handled. After the completion of sale, the inventory quantities should be updated.

Box 1 - Simplified Case Study Description

At any moment of time a student can refer to similar case studies and problem solutions, go back to the requirements stage of this problem, or ask for feedback on the diagram they produced so far (the Check button). If the diagram is correct, students can move onto the next methodological step to produce a system sequence diagram.

System sequence diagram editor suggests a list of events that could occur between the actor and the system. Again a diagram is displayed based on student’s selection of possible events. The example in Figure 3 shows that a student selected events enterItem, endSale and makeCashPayment, which involve candidate classes Cashier and the System, thus refining the Buy Items use case with a new event flow of three events.

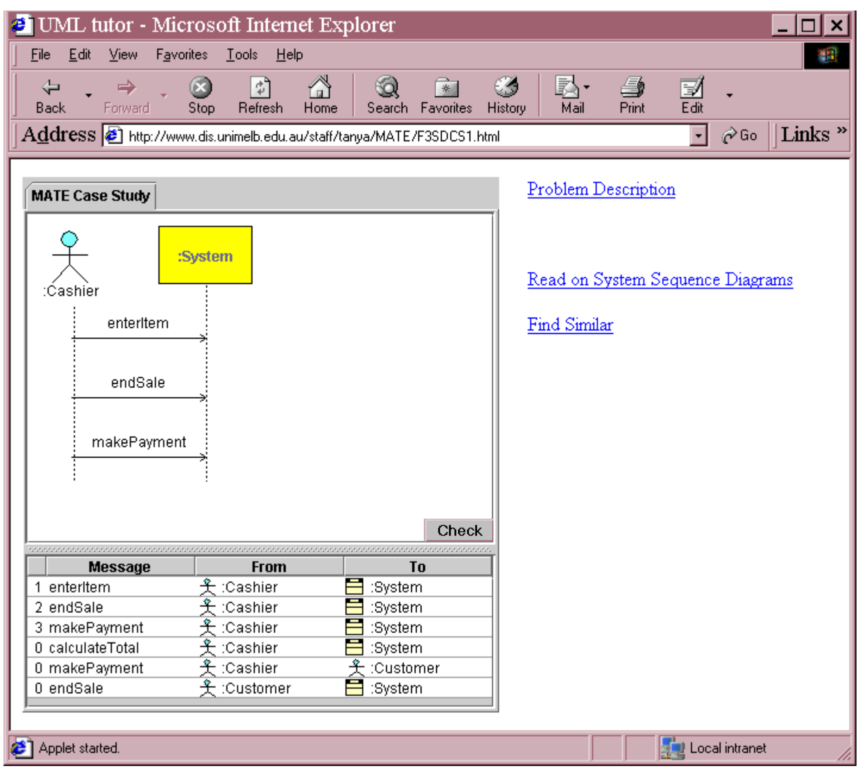

Figure 3 - System Sequence Diagram Editor

In the next step of the development process learnt by students, every event between actors and the system should be refined into either a collaboration diagram or an object sequence diagram. From the system sequence diagram shown in Figure 3, a student could produce three detailed object sequence diagrams for events enterItem, endSale and makeCashPayment.

In the process of creating required diagrams students apply appropriate design patterns. The system offers a list of patterns with their full description available on request. We will illustrate the solution development using the refinement of the *enterItem* as an example where the actor Cashier passes a message to the system *enterItem(barcode, quantity)*. The applicable pattern here is Controller, one of the GRASP[2] patterns. This pattern assists in assigning system events to components of the system.

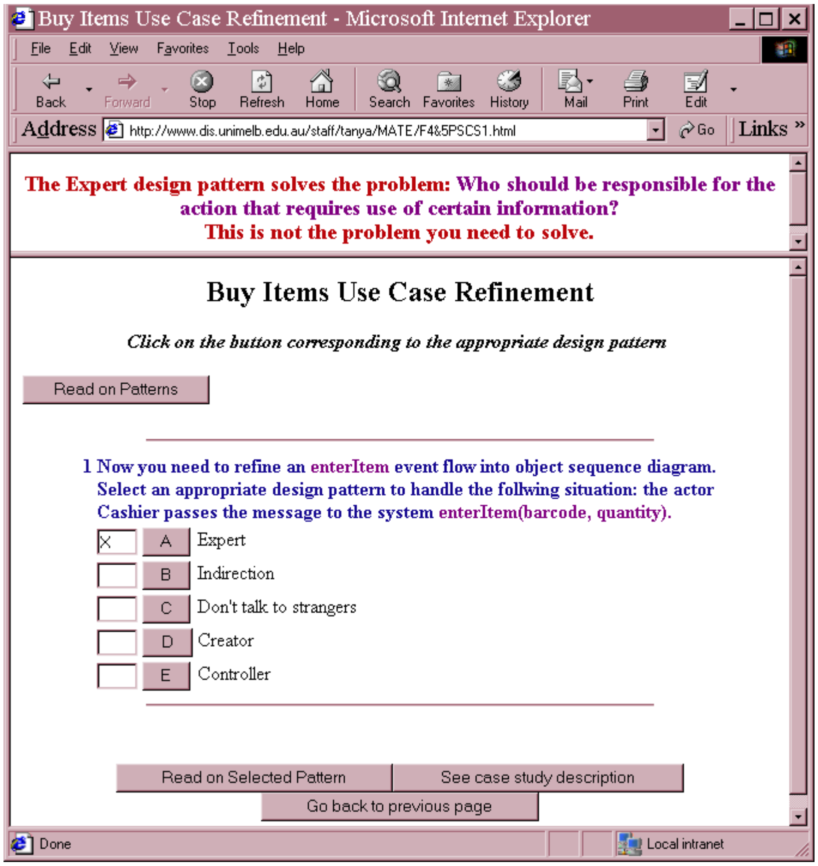

Figure 4 - Making Incorrect Choice

When a student clicks on the Check option, the system evaluates the diagram against the expected solution. If the produced diagram is correct the student is presented with the next task: he/she is offered one of the event flows for further refinement and a list of patterns to choose from that should assist in creating the sequence diagram. In this case, a student is solving the problem: what class or object is responsible for handling the enterItem system event? The expected pattern selection is Controller.

First lets see what happens when a student makes an incorrect selection, e.g. he or she selects the Expert pattern (Cf. Figure 4.). Expert pattern facilitates low coupling between classes by suggesting allocation of a method into a class that has the most information allowing it to fulfil the responsibility. The problem solved by the Expert resembles the problem currently addressed by the student, hence a mistake can be easily made. Nevertheless, the Expert is more appropriate for those cases when events are happening inside the system. In our case, we are talking about an

---

2 GRASP is a collection of pattern introduced by Larman in his textbook, it stands for General Responsibility Assignment Software Patterns

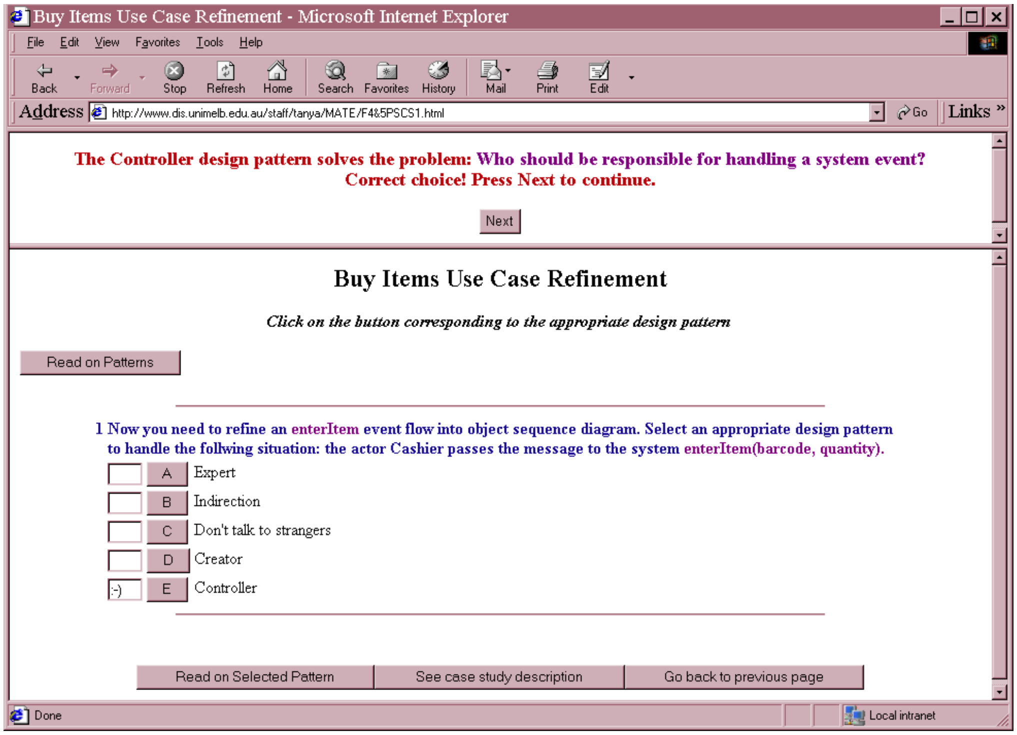

Figure 5 - Selecting Correct Design Pattern

event initiated by the actor outside the system, hence, we need to figure out which of the system component is able to handle the system-level event. To provide feedback, the student is shown a description of the problem associated with the selected pattern. The message suggests that this is not the problem the student should be solving.

When selected the correct answer, which is the Controller pattern in our example (Cf. Figure 5), the feedback includes the problem description associated with the pattern, but this time the student is told that the choice is correct and he/she could proceed with further refinement.

Based on the Controller pattern, four classes are candidates to handle this system event: POST, Store, BuyItemsHandler, and Cashier. These choices are displayed to the student: (Cf. Figure 6). For each selection the student gets an immediate feedback in the feedback frame (Cf. Table 1). For example, in case of the POST selection, we suggest that it is a good choice since there are a limited number of system operations to perform and we don't need to distribute

POST	This is façade controller - represents the overall system
Store	This is façade controller - represents the overall business
Cashier	This is role controller - represents something from the real world that may be actively involved with the task
BuyItemsHandler	This is use-case controller - represents an artificial handler of all system operations of a use case

Table 1 - Comments on the Selection of Candidate Objects in the Controller Pattern

responsibilities in order not to overload a controller class.

In the next step MATE presents a student with the part of the diagram constructed thus far and the next task is explained. At this step of diagram construction for a new customer a new Sale object should be created. Clicking on the Next button again displays a screen with the list of patterns and the student is expected to select the Creator pattern. This pattern helps identifying a class that will be responsible for creating a new instance of Sale. The solution suggests selecting the class that aggregates, contains or records the objects to be created. POST is a good candidate since it records details of a sale. Again for each correct or incorrect choice immediate feedback and discussion of the choice are available).

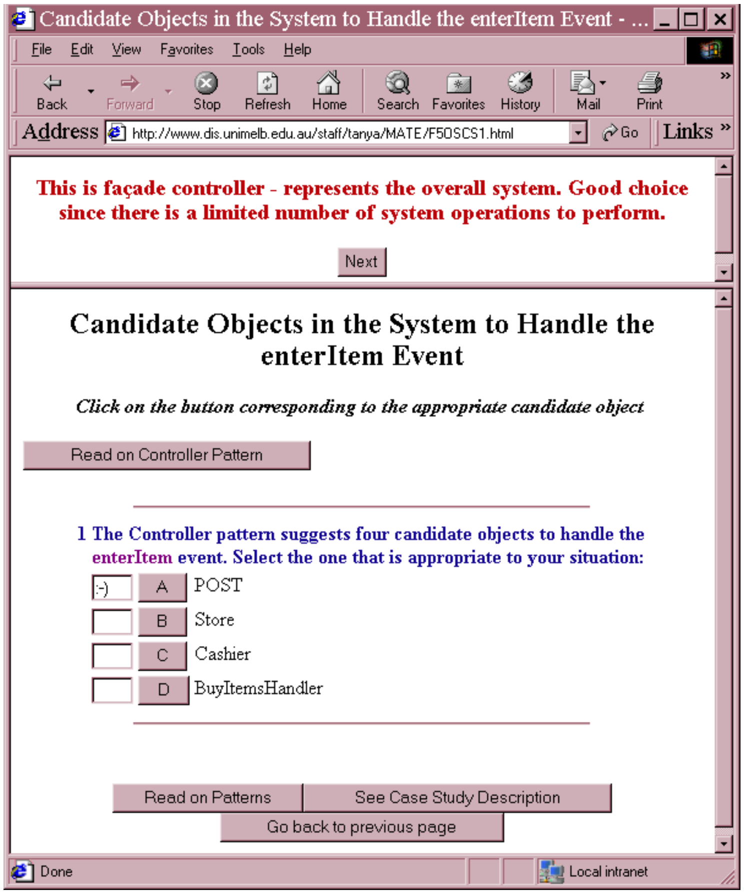

Figure 6 - Candidate Objects as per Controller Pattern

In this way students continue designing the solution under the guidance of the tool constructing sequence, collaboration, class and state diagrams. The system provides the students with some problem descriptions, engages them in snapshot quizzes which are an integral part of the case study, guides them towards high quality problem solution, explains its design decisions, and supplies the students with knowledge on good and bad design choices.

5. Summary and Conclusions

In the past, it has been indicated that there is a need for courses that are professionally designed and evaluated, which adopt effective learning strategies and encourage a self-directed learning [5]. We address these issues by teaching SA&D with the use of interactive multimedia learning environment MATE. In this paper we described MATE from the student's perspective. Our experience shows that employment of MATE complements traditional teaching with the use of lectures, tutorials and practical sessions. Use of MATE offers the following benefits:

- **Effectiveness:** students can work on case studies in their own time and at their own pace to suit their individual needs.
- **Flexibility:** lecture slides, case studies and solutions are available on demand, which provides wider choices of learning paths and individualises instruction.

- **Consistency:** the use of the MATE environment provides a source of standards that can be used in various teaching modes throughout the common repository of multimedia components.
- **Instructional needs:** the environment is designed to allow addressing specific objectives of the course through the use of case studies, creation of lecture slides, self-paced learning and assessment.
- **Links and information exchange between subjects:** case studies and teaching components from the multimedia repository can be used across the curriculum of related subjects.
- **Use of advanced design techniques:** various design approaches including UML and patterns are offered to students via displayed information, analysed case studies, ready-made solutions and guided solutions development.

Our main strategy was the use of a problem-solving approach. Through the use of case studies students develop good understanding of an object-oriented design process, the relationship between individual stages of the process, and most importantly they learn to judge the value of alternative ways to solving a problem in systems analysis and design.

6. Bibliography

1. Bearne, M., S. Jones, and J. Sapsford-Francis (1994): Towards Usability Guidelines for Multimedia Systems. in Second ACM International Conference on Multimedia '94. San Francisco, CA, USA, p. 105.
2. Forman, D. (1995): The Use of Multimedia Technology for Training in Business and Industry. Multimedia Monitor(July): p. 22-27.
3. Frankhauser, R. and H. Lopaczuk (1996): Exploring the Multimedia Landscape from a Training and Professional Development Perspective, in Learning Technologies: Prospects and Pathways, J. Hedberg, J. Steele, and S. McNamara, Editors. AJET Publications: Canberra, Australia.
4. Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1995): Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley.
5. Hosie, P. (1993): Technologically Mediated Learning: The Future of Training in Australia. Australian Journal of Educational Technology. **9**(1): p. 69-86.
6. Jacobson, I., G. Booch, and J. Rumbaugh (1999): The Unified Software Development Process. Reading, MA, USA: Addison Wesley Longman, Inc.
7. Keeler, C.M. and R. Anson (1995): An Assessment of Cooperative Learning Used for Basic Computer Skills Instruction in the College Classroom. Educational Computing Research. **12**(4): p. 379-393.
8. Kelly, A.E. and A.O. Donnell (1994): Hypertext and the Study Strategies of Preservice Teachers: Issues in Instructional hypertext Design. Educational Computing Research. **10**(4): p. 373-387.
9. Keppel, M.J. and J. Richards (1996): Choosing Multimedia as a Training Option: Client Considerations. in Fifth Annual Multimedia in Education and Industry Conference. Charleston, South Carolina, USA, p. .
10. Najjar, L.J. (1996): Multimedia Information and Learning. Educational Multimedia and Hypermedia. **5**(2): p. 129-150.

11. Reed, W.M., D.J. Ayersman, and M. Liu (1996): The Effects of Students' Computer-Based Prior Experiences and Instructional Exposures on the Application of Hypermedia-Related Mental Models. Educational Computing Research. **14**(2): p. 185-207.

12. Shin, Y.-F. and S.M. Alessi (1996): Effects of Text versus Voice on Learning in Multimedia Courseware. Journal of Educational Multimedia and Hypermedia. **5**(2): p. 203-218.