A Scaffolded Approach to Teaching Information Systems Design

Dr. Steve Wade
University of Huddersfield, UK
s.j.wade@hud.ac.uk

Mohammed Salahat
University of Huddersfield, UK
m.salahat@hud.ac.uk &
Ajman University, UAE
m.salahat@ajman.ac.ae

Dr. Dave Wilson
University of Huddersfield, UK
d.r.wilson@hud.ac.uk

ABSTRACT

This paper reflects on the experience of delivering a module in Information Systems Design to postgraduate students. The module has been taught for a number of years but has recently been restructured around a novel systems development framework presented as a pattern language. This restructuring represents a move towards a "scaffolded" approach to delivering the module. We present evidence that this approach has improved both the students' technical skills and their confidence in applying these skills.

.

Keywords

Scaffolding, postgraduate, information systems design, pattern language.

1. INTRODUCTION

The research presented here is a descriptive case study based on our experience of delivering a postgraduate module in Information Systems Design to students of our MSc courses in Advanced Computer Science and Information Systems Management. For a number of years this module has been taught in block mode over five full days. We chose this mode of delivery to attract part-time students in full-time employment. Over the years the profile of students on the courses has changed from predominantly working adults to predominantly full time international students. It became apparent that the intensive nature of block week teaching caused difficulties for this latter group of students who often arrive for the first time in the UK just a few days before their first class. Restructuring the module to be delivered over a full semester to full-time students presented an opportunity to rethink the modes of delivery and assessment. We have now adopted a "scaffolded" approach based on a systems development framework that we have developed and applied over a number of years (Salahat and Wade, 2009; Salahat *et al*, 2009).

This paper will explain the nature of "scaffolded" teaching as we have applied it in our "Information Systems Development" module. We first provide a brief introduction to the idea of scaffolding in teaching, then comment on our research methodology and the structure of the development framework we have followed, we then explain how we have evaluated that framework through our teaching. The paper is concluded with a brief description of our plans for further research in this area.

2. USE OF SCAFFOLDING IN TEACHING

The term "scaffolding" was introduced by Bruner (1966) to describe a framework of supportive elements added to a teaching programme to help students develop a deeper understanding of a subject. The approach encourages support for various learning styles and learning experiences (Salend, 2001; Kame'enui et al., 2002; Kirk et al, 2006;) by "actively diagnosing student needs and understandings, providing tailored assistance and specific feedback" (Larkin, 2002, p.30). This means that the tutor has to be able to identify an area that is just beyond, but not too far beyond, the student's current understanding. The approach is rooted in constructivist theories of learning which emphasise the active role learners take in constructing their own individual knowledge schemas (Duffy and Jonassen (1992), p.64). The challenge is how to assist students to

make links between new knowledge and what is in their existing schemas (Ryan and Carroll, 2005, p.14). This can be difficult with a diverse group of students such as the ones we have been teaching some of whom have a background in Computer Science whilst others have a background in Business Studies.

The Information Systems Design module discussed in this paper is based around the application of the Unified Modelling Language (UML) throughout the development lifecycle from requirements analysis to implementation. All of our students arrive on the module with some background in modelling but those on the MSc Advanced Computer Science course tend to view modelling as high-level programming whereas those studying for MSc Information Systems Management tend to think in terms of business models. This presents the challenge of moving students into a deeper understanding from different starting points and with different preconceptions about the nature of the subject.

In later sections we describe how we have built our basic scaffold around a framework of techniques that help the developer to move step-by-step from modelling human activity to implementing a software system to support that human activity. We will also explain how we have used a number of feedback mechanisms to encourage students to reflect on their learning at each step along the way. Before this we will comment on the basic methodology we have been following.

3. METHODOLOGY

The work reported here is an action research project aimed at improving educational delivery on one module. A variety of forms of action research have been proposed in the context of higher education (McPherson and Nunes, 2004). In a typical action research project the researcher will occupy two roles: one as the proponent of an educational theory and the other as a user of that theory. The typical action research project will be based on an iterative lifecycle embracing problem identification, action planning, implementation, evaluation, and reflection. The insights gained from an initial cycle feed into planning of the second cycle for which the action plan is modified and the research process repeated. We have reached the stage of completing the first cycle which has included the following activities each of which will be described in more detail later in the paper:

- The identification of key learning outcomes and initial working hypotheses about how to meet them. For instance, a key objective is to provide an appropriate framework for exploring the expert knowledge contained within systems development methods. In developing this framework we were mindful of the need to encourage maximum student ownership of the learning process. The framework cannot therefore be a simple list of instructions to be followed slavishly. We have therefore presented the framework as a pattern language to be used by students to help them generate their own high-quality software designs.

- The development of teaching materials to document the framework. This has been done by developing a website containing descriptions of the various patterns that make up the framework and the links between them that make this a pattern language.

- Running the module. The module is based upon a one hour lecture and two hours of project work per week over one semester. The lectures have been made available as ScreenCam videos *via* the University's video-streaming service. Students are expected to view these ahead of the actual presentation; this helps to make the lecture sessions more interactive as students arrive with a prepared mind. Students are encouraged to maintain a learning diary in the form of a blog within Blackboard, the University's Virtual Learning Environment. The blogs are visible to other students as well as the tutors.

- Observing the progress of the module week-by-week in a number of ways including a range of on-going student feedback mechanisms.

- Reflecting upon the results of the evaluation, in preparation for modifying delivery for the second presentation of the module.

With respect to this final activity we used a number of different ways to collect information as the basis for reflection; a pre-course questionnaire was distributed before teaching began to establish the background knowledge and preconceptions of our students; a series of anonymous in-class surveys were used to test students understanding and self-confidence in the techniques being introduced; an analysis was carried out of the most common mistakes made by students in the models they submitted as part of a coursework portfolio; short reflective essays were made part of the coursework portfolio in which students were asked to comment on the perceived benefits and disadvantages of the techniques; questionnaires were administered to give the students the opportunity to feedback on any issues they did not feel had been covered elsewhere and focus group discussions were held in class. The move from teaching in block mode to a semester-long presentation meant that data could be collected, but it had to be analysed quickly in order for changes to be made to the module from week to week.

4. THE DEVELOPMENT FRAMEWORK

The development framework we have followed in teaching this module is based on our earlier research into the design of a multi-method framework (Salahat *et al*, 2009). In that project we proposed a framework for bringing together principles from object oriented approaches to designing software systems and the "soft systems" approach to analysing social systems (Checkland, 1999). Following the basic structure of this framework we developed a module based around the following topics:

- How to use soft systems methodology to learn about a problem situation.
- How to extract Use Cases from the soft systems models.
- How to develop sequence diagrams related to each Use Case.
- How to develop a domain model from the collection of sequence diagrams.
- How to convert the domain model into a class diagram and database design.
- How to implement the class diagram as an object oriented software system using the, "naked objects", implementation pattern.

We have used the basic step-by-step approach suggested by these topics as the basis for scaffolded teaching. Initially we present a number of complete case studies following the steps implied above then ask students to work on related case studies for their coursework. The case studies are based around the needs of an academic department like our own.

It is beyond the scope of this paper to discuss the details of each of these topics or present the case studies that have been developed but for those unfamiliar with the techniques alluded to the following examples are intended to give a flavor of the deliverables developed during each step of the method.

The example used here relates to the decision to introduce a Peer Tutoring System into an academic department to provide extra help to students on a programming module. We would initially begin by drawing a rich picture. The diagram below shows a rich picture produced in a recent tutorial.

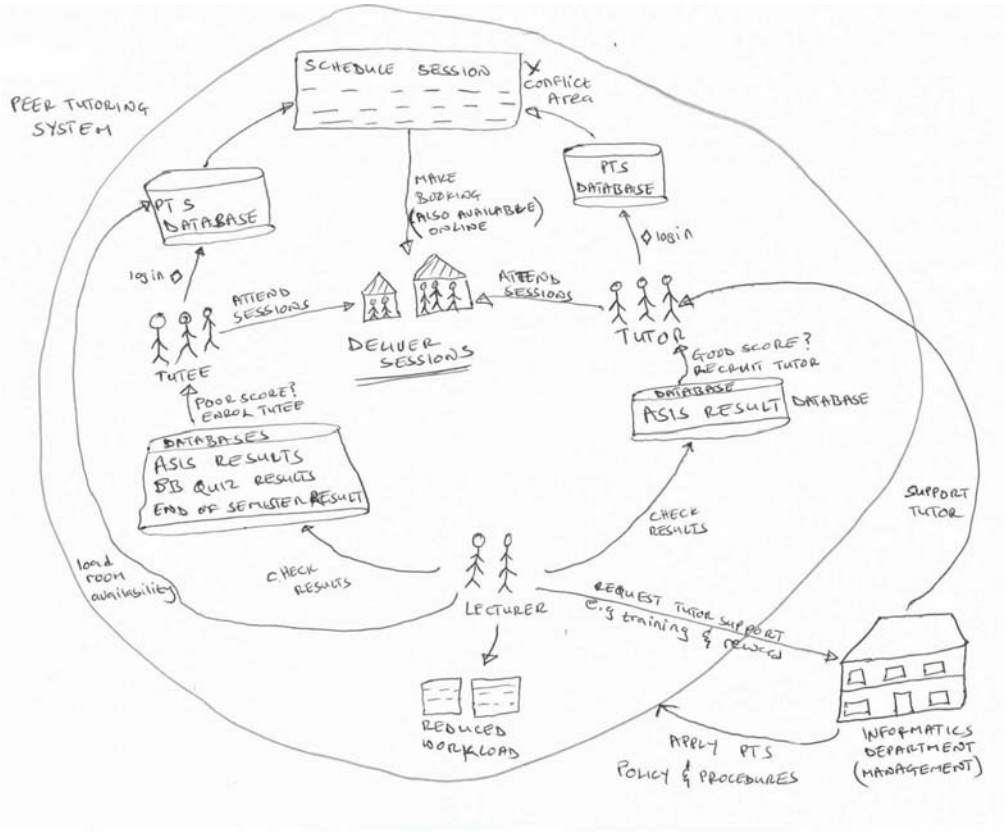

Figure 1. A rich picture for the Peer Tutoring System.

The use of rich pictures with an international group of students is a good way to start. Simple graphics are understood across language barriers and are fun to develop and discuss. The discussion stimulated by the rich picture leads us into developing a root definition: A succinct description of the system being developed. The following might be a suitable root definition for the peer tutoring system:

- A system owned by the school that provides study skills support to students using volunteers from the student body with the quality of their support activities monitored by academic staff.

Having agreed a root definition we move to developing more formal activity models such as the following which includes activities that might be supported by a software system along with others that will be enacted by humans without the assistance of software.

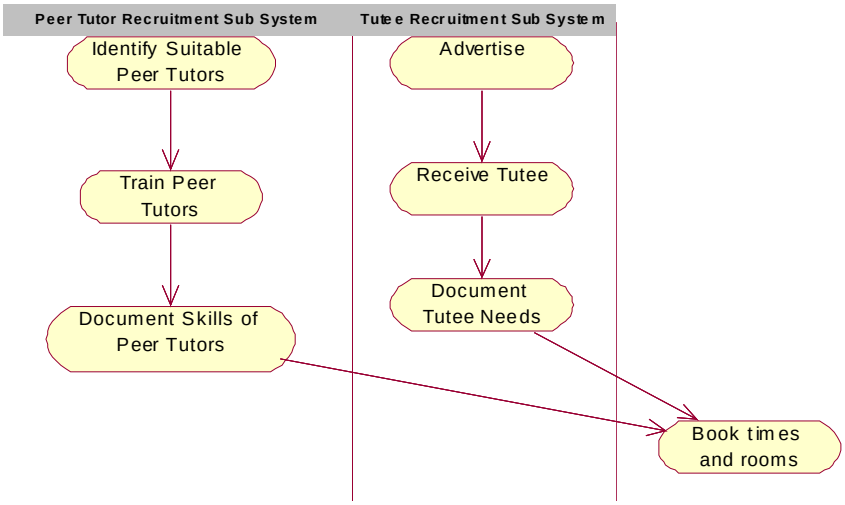

Figure 2. Activity Diagram.

In tutorials we use these diagrams to provoke further discussion. For example we might consider the following questions:

- Is it enough to advertise the peer tutoring service or should some (weaker) students be required to attend?
- Should we pay peer tutors?
- How should the effectiveness of the system be measured?
- Should we monitor the attendance of students at these sessions?

Some of these questions lead us to develop further activity models. For example the following for an "attendance monitoring" system:

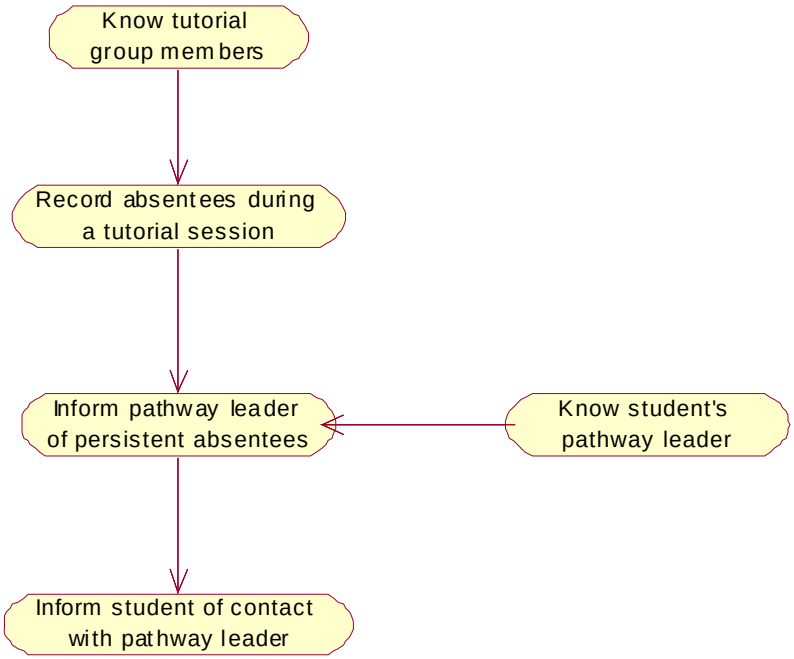

Figure 3. Activity Diagram for Attendance Monitoring.

These types of diagram are intended to describe human activity but they can be used to inform the design of a Use Case Model. Use Cases are activities that require software support. The following diagram could be derived from the activity diagram above:

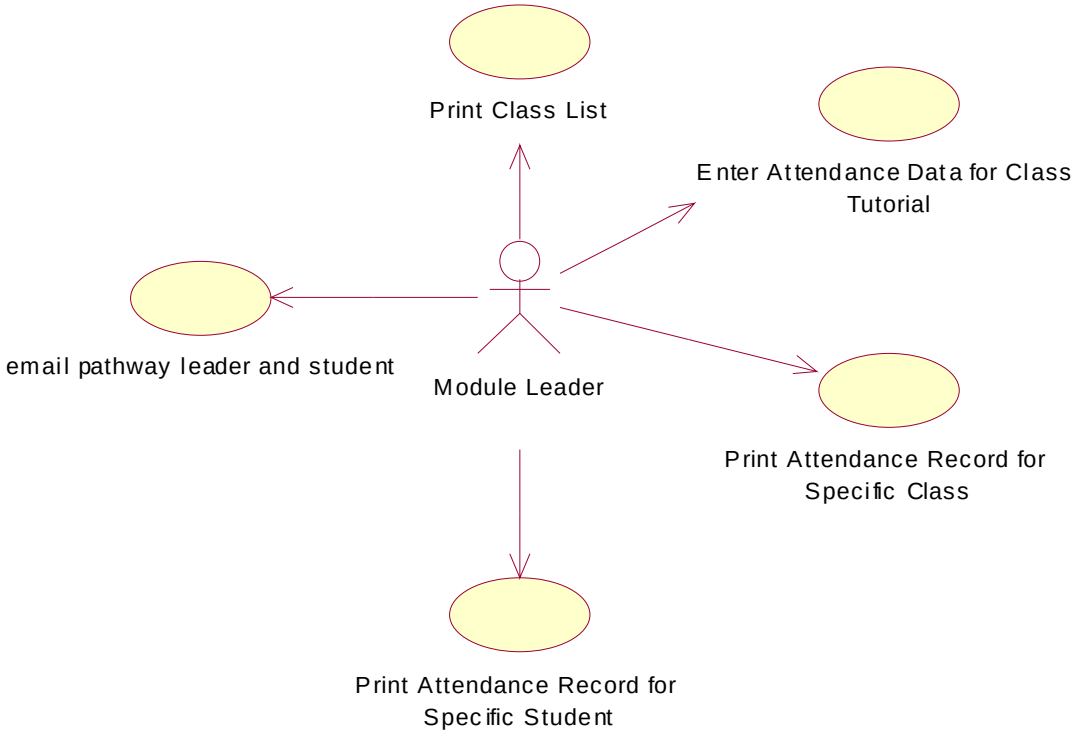

Figure 4. A Use Case Model.

If we focus on the "Print Class List" Use Case we might develop the following user interface in which the user enters a Module Code along with details of everyone who should be attending the peer tutorials for that module.

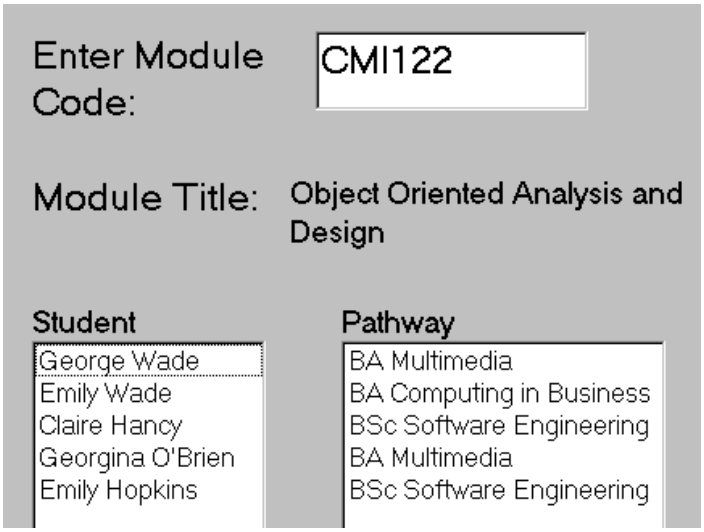

Figure 5. Screenshot for a Use Case.

To support this interface we could propose a sequence diagram explaining the role that a number of objects will have, "…behind the scenes".

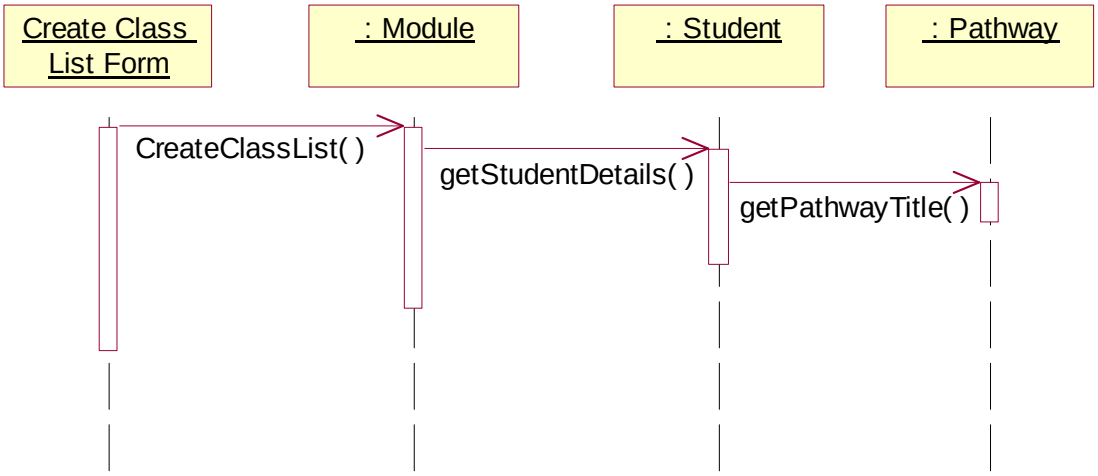

Figure 6. A sequence Diagram.

We would develop a diagram like this for every use case then develop a domain model consistent with all of these diagrams. A domain model derived from this one sequence diagram might look like this:

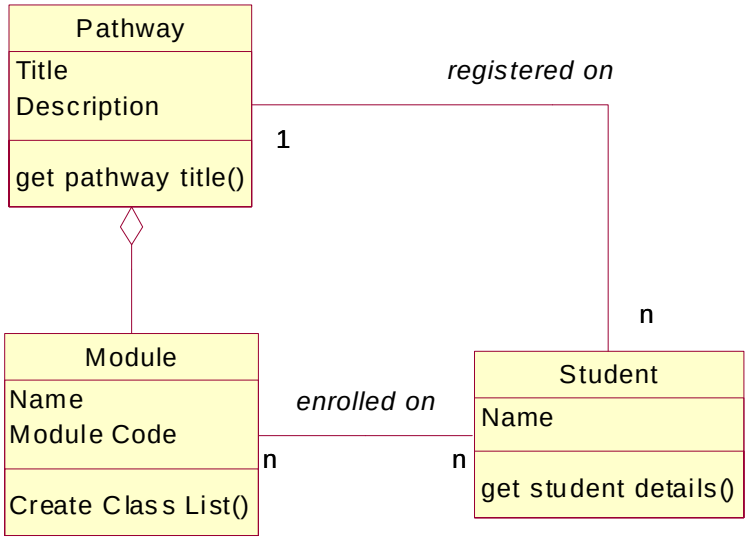

Figure 7. A Domain Model.

The domain model can be used as the basis for designing an object oriented software system and a relational database structure. We have encouraged students to use the Naked Objects Framework (2012) to generate object oriented code and a graphic user interface directly from the domain model. The user of an application developed using the Naked Objects Pattern interacts with a series of windows containing icons representing each of the domain classes. A class can be accessed by double-clicking on its icon to reveal its attributes and operations in a standard format. In the above example I can select, “Module,” then search for a specific module and right-click on its “Create Class List” operation to see a list of students enrolled on the module. Other functionality can be achieved by dragging and dropping; so for example if I wish to enroll a specific student onto a specific module I can drag the icon representing that student on to the icon representing the module thus creating the relationship between them. This means that the relationship between the code and the model is very literal. A change in the model (e.g. the addition of an operation on, “Student”, named “Get Coursework Marks”) feeds through into the code and then directly into the user interface. From a teaching perspective this helps to reinforce the idea that modeling is both about representing the real world and designing software.

5. PATTERNS IN THE DEVELOPMENT FRAMEWORK

An important part of our research has been to identify specific issues that cause difficulties for students then provide specific, detailed guidance of how to ameliorate these difficulties. We have done this by specifying development patterns that can be applied at each stage.

Patterns have been widely used in information systems design over the last ten years. A pattern in this context is a generic solution to a recurring problem expressed in a literary form. The approach has its roots in architecture specifically the work of Alexander (1979). In information systems design patterns have been used to ease communication problems and the thinking behind complex design. Patterns are usually described by templates which specify the style and structure of a pattern description. Typically the template will include sections for a description of the problem to be addressed, the forces acting to create the problem, a generic solution, a specific example of how this solution might be applied and a discussion of the benefits the solution should provide.

The following example relates to a common problem in domain modeling where students represent a many-to-many relationship between two objects when the relationship would be better represented by a third object.

Problem:
How to model the relationship between two classes that have a many-to-many association with each other?

Forces

- Many-to-many relationships occur often in the real world.
- It can be difficult to implement many-to-many associations in some object oriented programming languages.
- Many-to-many relationships have no direct implementation in relational database systems in which they may have to be persisted.
- A many-to-many relationship is usually complicated enough to warrant the addition of an extra class.

Solution
Transform the many-to-many association between two classes into a trio of classes by creating an intermediary class with two one-to-many relationships. The name of the intermediary class should describe the type of relationship being captured.

Example

A many-to-many relationship between Drug and Patient is reconstructed as two one-to-many relationships. One between Drug and Prescription, while the other between Patient and Prescription.

Discussion

- We can now store details of date and dosage for each prescription as attributes of the new “prescription” class.
- Prescription might be linked to Doctor to identify the individual doctor who made each prescription.

The idea is that patterns such as this can be used to guide students away from common problems and into good practice. We have tried to identify these common problems by looking for recurring mistakes in coursework. For example students found the transition from Use Case Models to Sequence Diagrams difficult so we provided the following pattern and discussed it in class.

Problem:

It is hard to develop sequence diagrams from the Use Case Module. What can I do to make this transition easier?

Forces

A high level Use Case Diagram (such as the one presented above) is fine for a, "mile high", view of the computer systems behaviour. For many stakeholders, such as sponsors and managers, this will be enough. As designers however we need to open these up and define them in detail. We know what the system presents to the various

users (or actors), we need to define in fine detail the, "how", of that interaction; until we have done this we cannot begin to develop a sequence diagram.

Solution

Using a set format for the Use Cases makes the collection and organisation straightforward. The following format for a Use Case Pro-forma is suggested:

Use Case Number:	Use Case Name:
Goal:	
Brief Description:	
Actors:	
Quality requirements:	
Primary Path:	
Use Cases Related to Primary Path:	
Alternatives:	
Use Cases Related to Alternatives:	
Exceptions:	
Use Cases Related to Exceptions:	
Notes:	

Figure 9. A Use Case Template

The main section of this pro-forma is the one describing the primary path – this is a step-by-step description of the way in which the use case will be executed once the software system has been developed. In specifying the primary path we assume that nothing will go wrong during any of these steps. Any problems that might arise are dealt with separately in the Alternatives and Exceptions sections. We can use the description of the primary path as the basis for developing an initial sequence diagram. The way in which this is done is specified in another pattern.

There are some additional fields in the pro-forma that will not be discussed here.

Example

Use Case Number: 1	**Use Case Name:** Enroll in Peer Tutor Session
Brief Description: This Use Case is concerned with enrolling an existing student in a peer tutor session for which she is eligible.	
Actors: Module Leader; Student	
Frequency of Execution: Daily	
Scalability: Only one instance of this runs at any one time.	
Criticality: Essential. We need an accurate record of who is expected to attend each session.	
Precondition: The student is registered at the university and is currently studying the module related to this peer-tutor session.	
Postcondition: The student will be enrolled on the peer tutoring session if she is eligible and there are spaces available.	
Primary Path: 1. The use case begins when a student wants to enroll in peer-tutor session. 2. The student inputs her name and student number into the system. 3. The system verifies the student is eligible to enroll in classes at the university 4. The system displays the list of available peer tutor sessions 5. The student indicates the session in which she wishes to enroll. 6. The system checks that the student is enrolled on the appropriate module to join in the session 7. The system asks the student to confirm that she wants to enroll in the session. 8. The student indicates she wants to enroll in the session. 9. The system enrolls the student in the session.	
Use Cases Related to Primary Path: None.	
Alternatives: The student is not eligible to join classes. The student is not studying the appropriate module for this session.	
Use Cases Related to Alternatives:	

Figure 10. A Use Case Example.

The above are examples of two patterns we use in teaching. We have specified many more patterns related to commonly occurring problems. Initially we used patterns drawn from the publications of Ambler (1998), and Evitts (2000). We spent some time re-working and shaping the documentation for these patterns to give coherence to the collection. The patterns are presented in a website based around the metaphor of different development, “rooms”. The first room is concerned with developing a soft systems model and contains

patterns for developing a range of soft systems models including a rich picture, root definition and conceptual models. An adjoining room contains patterns for translating the conceptual models (more specifically the "consensus primary task model") into a Use Case Model with detailed documentation of each use case. A door in this room leads to another containing patterns that explain how to:

- Develop a sequence diagram to show how the behaviour of a use case is accomplished by objects. This will involve ensuring that the use case primary path maps across to the messages being passed on the sequence diagram.
- Assign operations to classes that map to messages on the sequence diagram.

When a number of patterns are related to each other and have a common domain we describe this as a, "pattern language". We are therefore trying to develop a pattern language to support the teaching of information systems design. We would argue that patterns are particularly suited to this purpose. They are descriptive, not prescriptive (unlike most development methods). They capture expertise in a loose and open-ended format that lends itself to a "hypertextual" presentation in the form of a website with links between related patterns that do not force a specific sequence of activities. The patterns can also be used as the basis for developing class tests and more substantial assignment specifications.

6. PROPOSED MODULE FRAMEWORK

In light of the above discussion we have been able to propose the following guidelines for developing a scaffolded module in this area:

1. Design a portfolio-based assessment that can be completed step-by-step and is aligned to patterns used in teaching. For each pattern we specify outcomes that can be represented in an assessment grid. The pattern then becomes part of the explanation of what is required and is clearly linked to the feedback grid.

2. Provide formative in-class surveys that encourage students to reflect on their understanding of key patterns. Do they understand why something has been identified as a problem? Can they see how the proposed solution would help? Can they apply it to the coursework case studies?

3. Encourage students to discuss the individual patterns and how they may be applied to case studies in groups before they complete the in-class surveys. Students should also work on parts of the coursework that relate to specific patterns on a week by week basis.

4. Collect data on a regular basis by inspecting student coursework and in-class surveys. Use the feedback to inform improvements to pattern descriptions and the identification of new patterns.

These four steps work together to support the students through the assessment process by constantly monitoring their progress and providing support that helps them to identify what they are expected to do and how. The feedback mechanisms lead to dynamically updating and refreshing the module content. Hopefully this will lead to continuous improvement in the clarity of the teaching materials.

7. EVALUATION

7.1 Pre-course Questionnaire

Thirty eight students joined the Information Systems Design module in 2011. A background questionnaire was distributed to them before the first class to gather information about their prior learning in this area. An analysis of the questionnaire shows that there were broadly two types of student taking the module:

- 18 students of MSc Advanced Computer Science. These students had a strong background in programming. Some experience of modelling but not with the UML. None of them were familiar with

the idea of multimethodology. None of them had heard of SSM. Whilst studying our module these students were also studying advanced software development modules in areas such as, “internet application development”.

- 20 students of MSc Information Systems Management. These students do not have a strong back ground in programming. Most of them were unfamiliar with the principals of object oriented programming. Some experience of modelling but not with the UML. Most of them had heard of SSM but were not aware of the literature on multimethodology. Whilst studying methods and modelling these students were also studying information systems modules in areas such as, “competing in a digital economy”.

The module finished in December 2011 and the student’s feedback has been generally positive. There is a clear indication that this new way of presenting the module has led to an increase in the overall pass rate and higher grades for students.

7.2 Reflective essays

For the final part of the coursework portfolio students were asked to write a reflective essay including a discussion on how the module reinforced (or otherwise) their appreciation of the techniques and processes employed in undertaking a development project. In addition the evaluation had to include a wider discussion on topics such as:

- How well the module relates to the other modules on their course.
- How the knowledge and skills taught on the module relates to their previous experience as a student and/or employee.
- The appropriateness of the knowledge and skills taught on the module for future employment.
- Any particular aspects of the module that they found difficult. Specifically any aspect of the real world that they wanted to capture in the models that they developed or any steps in the process that seemed to be a waste of time. Or any additional steps that they thought might have been useful.

These essays provided generally positive feedback. It is possible that some students gave positive comments out of good manners or the mistaken belief that this might lead to higher marks. We sought to minimise this source of bias by making it clear that the students’ objective evaluation was important to us as part of our action research project. We explained that the patterns are interesting only in so far as they can be used by human beings to generate useful design ideas. If students considered them to be too vague, too time-consuming or unhelpful for any other reason we needed them to report this. The following comments are representative of some of the more general comments made in these essays:

- All of the techniques have proved very useful for me. I know how to design systems properly now.
- I have learned a lot from working in groups and following the method. I think this is the most important module because it links everything together.
- Before I started the module I did not know what modelling was or how it related to programming. I feel confident now that I can apply the techniques we have looked at on a real project.

Certain generalisations about the two groups can be made:

- The MSc Advanced Computer Science students were more comfortable with abstraction in the sequence and class diagrams. They seemed to regard modelling as high-level programming.
- MSc Information Systems Management students were more comfortable seeing sequence diagrams and class diagrams as models of the real world.

In future presentations of the module we propose to create mixed groups so that each student gets to work with students on a different course.

7.3 Analysis of the common mistakes in the class work

An analysis of the coursework submitted by the students revealed a number of common mistakes. A list of common errors would include the following:

- Failure to use domain-specific terminology as presented in case study materials.
- Inconsistencies between sequence diagram and class diagram. For example operations appearing in the sequence diagram that are not present in the class diagram.
- Operations given ambiguous names.
- Operations not supported by attributes or relationships.
- Database concepts (pk and fk) used in domain model.
- A lack of consistency between the SSM models and the Use Case Model.

We are working on developing patterns that will steer future students away from making these types of mistake.

7.4 In-class surveys

We used frequent in-class surveys to evaluate student satisfaction on a week-by-week basis. From these it was apparent that our focus on identifying patterns to help students through difficult techniques was helpful. The majority of the students (approximately 60%) claimed no prior experience of developing business models but after completing the module, 86% said they felt confident with the use of Soft Systems techniques. There was 100% agreement that the ongoing feedback provided in this module was very useful. Typical comments included:

- I like the step-by-step approach where we move forward slowly with help at each stage. I think I would have become confused if I had to do all the work at the end.

- It helps to chunk up the work with patterns. Each pattern seems to make sense and when you put them all together you can make something happen.

As teachers we found that the approach taken was very time-consuming and might be difficult to implement when working with larger groups. Our focus on ways in which we could develop pattern-based teaching materials did lead us to spend more time looking at the students' work than we might otherwise have done. This helped us to see more clearly what techniques the students found hard to understand.

8. FURTHER WORK

We are developing a website for this and similar modules. This will comprise the collection of patterns we have used with hyperlinks between related patterns. For each pattern we will provide a worksheet that specifies a marking scheme to confirm that each step in the pattern has been followed. A number of students have used our pattern language in their dissertation so we are also building up a repository of relevant case study materials.

Where possible we have applied the concept of a pattern language beyond the boundaries of the specific module we have been discussing. We have applied the basic pattern template to topics covered in other modules and this provides a mechanism for highlighting relationships between modules. For example on a project management module we consider a case study related to the activities of our own department. The documentation for this case study is presented as a series of patterns following our format. A partial example is provided below:

Problem: Short-term Absence of Staff.

Context: The short-term (less than a month) absence of staff affects the work processes.

Forces:

- Absence because of illness, injuries, professional training, holidays or weather.
- Administrative tasks are interrelated in complex ways relying on staff expertise.

Solution:

- Provide online or paper handbook of instructions for specified routine tasks of each member's duties.
- Urgent work should be picked up by another member of staff as soon as possible.

Discussion:

There should be online or paper handbook of instructions for specified routine tasks of each member's duties that would help in the period of their absences. Online or paper instructions must be left behind by staff (admin) on leave especially during busy times e.g. May to September, in order to assist replacement/duty staff. This will help duty staff in covering absentee's jobs.

We also use design patterns as discussed in Gamma et al (1995) in all of our software engineering modules. It is hoped that the use of the pattern concept across a range of modules will create a sense of coherence and integration across the course as whole.

9. CONCLUSION

This paper has reviewed our experience of delivering an Information Systems Development module to a postgraduate, largely international, group of students. We have described our presentation of the module as a, "scaffolded", approach. The scaffolding in question has been built around a novel systems development framework presented in the form of a pattern language. This basic structure has been fleshed out with a number of feedback mechanisms (including in-class surveys, feedback questionnaires, focus group discussions and reflective essays) and a sympathetic assessment strategy. We have concluded that the approach yielded significant benefits for the one module discussed here but might also have wider applicability.

10. BIBLIOGRAPHY

Alexander, C. (1979). *A Timeless Way of Building.* New York, NY: Oxford University Press.

Bruner, J.S. (1966) *Toward a theory of instruction*, The Belknap Press of Harvard University Press, Cambridge, Massachusetts.

Ambler, SW. (1998). Software process patterns, Cambridge University Press

Ambler, SW. (1999). More software process patterns, Cambridge University Press

Checkland, P. (1999) *Soft systems methodology: a 30-year retrospective,* John Wiley, Chichester.

Duffy, T. and Jonassen, D. (1992) *Constructivism and the Technology of Instruction: A Conversation,* Routledge, London.

Evitts, P. (2000), *A UML pattern language,* Macmillan Technical, Indianapolis, Ind.

Gamma, E; Helm, R., Johnson, R. and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, MA: Addison Wesley.

Kame'enui, E. J., Camine, D. W., Dixon, R. C., Simmons, D. C., and Coyne, M. D. (2002). *Effective teaching strategies that accommodate diverse learners* (2nd ed*.,* Merrill Prentice Hall, Upper Saddle River, NJ. Kirk et al, 2006;)

Kirk, S. A., Gallagher, J. J., Anastasiow, N. J., and Coleman, M. R. (2006) Educating Exceptional Children (11th ed.), Houghton Mifflin, Boston.

Larkin, M. J. (2002). *Using scaffolded instruction to optimize learning.* Washington, DC. (ERIC Document Reproduction Service No E639).

McPherson, M. and M. Nunes (2004). *Developing Innovation in Online Learning: an Action Research Framework.* London: Routledge Falmer.

Naked Objects (2012) *Naked Objects Framework - Product description* [online] available at: <http://nakedobjects.net/product/product_intro.shtml> [accessed on 3rd March 2012].

Ryan and Carroll (2005) *Teaching international students: improving learning for all.* Routledge, London

Salend, S. J. (2001) *Creating inclusive classrooms: Effective and reflective practices (4th ed.)*, Prentice Hall, Upper Saddle River, NJ, Merrill.

Salahat, M., Wade S.(2009) A Systems Thinking Approach to Domain-Driven Design, In the *proceeding of UKAIS2009 conference, Oxford University, Oxford, UK*.

Salahat , M., Wade, S., Ul-Haq, I.(2009) The Application of A systemic Soft Domain Driven Design Framework, WASET *online Journal, Issue57,pp476-486*.