

Case Study

LTSN Engineering Case Study: Developing the Independent Learner

May 2004



Authorship

This report was commissioned by LTSN Engineering and was written by:

- Alan Robinson, Faculty of Technology, Southampton Institute
- Mark Udall, Faculty of Technology, Southampton Institute

Edited by LTSN Engineering staff.

Published by LTSN Engineering
ISBN 978-1-904804-17-8
© LTSN Engineering 2004

1	Background	1
2	Methodology	2
3	Issues	3
4	Benefits	3
5	Evidence of success	4
6	The Mexican Hat Approach.....	4
7	Reflections.....	5
8	References	5

Abstract

Independent learning is taken to refer to activities that are driven by the learner's own enquiry. In order to develop this independence, the teaching and assessment strategy needs to be designed to encourage and develop the learner's ability to identify, structure and articulate questions about their own understanding. Formative assessment can play an extremely important role in this development process, if it explicitly places a focus on 'what learners do' rather than on 'what teachers do',¹ and thus breaks the perception that learning only takes place during formal timetabled sessions. However, the challenge is to achieve this within a manageable workload for both learners and teachers.

1 Background

The generic learning and teaching framework underpinning this case study has been refined and developed over a number of years. It is sufficiently straightforward to be easily understood by the students and also easily implemented, in a variety of ways depending on the subject discipline and level of study, by the teacher. The framework is characterized by:

- specific outcomes associated with each learning session,
- learning activities that allow students to assess their own learning and identify areas of uncertainty or problems, and
- a method that allows students and teachers to record the extent of participation and achievement in these learning activities.

The framework was first implemented on a level one unit teaching Pascal programming to students studying on a BEng Programme in Engineering Design. This case study presents a different implementation of the framework on a level two unit on the same programme called "Software Engineering".

In approaches where the formative feedback process is driven by the teacher there is the potential for also increasing the teacher's workload. This may be through additional marking and commenting on the work students generate in response to the formative activities. Whether students reflect on the feedback has also been questioned by researchers. In addition students are often averse to formative activities that are seen as just extra work and that do not 'count'. Therefore, the teacher's increased workload may not be particularly efficient or effective in promoting learning.

In the approach described in this case study the students generate the feedback. This not only promotes greater ownership by the students of their learning but it also enables the teacher to effectively target learning support according to the needs of the student. Being able to target the support required also makes the approach more efficient, which in turn reduces the teacher's workload.

Our experience was that some students fall into a cycle whereby if they find a unit difficult, they get behind, fail to attend, fail to seek help, become de-motivated and finally fail the unit of study. Therefore, a learning, teaching and assessment strategy was designed to account for such students, by identifying and supporting them, especially early in their studies, before they fell into this cycle.

A series of learning activities were designed to provide feedback so that the learner and teacher could determine progress towards the achievement of the learning outcomes and identify the need for any additional learning support. The assessment associated with the learning activities not only performed a diagnostic and formative role, but also incorporated a

¹ This is a rewording taken from Biggs 1999, "What the student does".

small summative component (this provided motivation for the students and explicitly linked effort with success) into each successive assessment. The combination of learning, teaching and assessment methods were designed so that throughout the unit students were assessed on the essential concepts on a number of occasions. Revisiting concepts and ideas on several occasions within assessment, supported by a formative dimension, promoted reinforcement, provided opportunities for each student to achieve their maximum potential, and subsequently improved performance. The aim was to make learning the goal rather than just passing.

2 Methodology

The “Software Engineering” unit had a reasonably traditional structure, comprising a single lecture (see Annex A) and a workshop each week, both of one hour's duration. The teaching approach was via case studies e.g. exploration of software engineering principles as applied to a bottling plant or modelling of a vending machine.

There were 19 students studying the unit part-time who were in their first year of the degree programme, having obtained direct entry to this level of study with a HNC/HND, plus 13 full-time students who had progressed from Level 1.

Each student was expected to maintain a portfolio as a record of the outcomes from the activities they undertook. This formed the basis for learner-teacher conversations about the learner's developing understanding and was used to provide validation/evidence of what the student had been doing.

Each student was also provided with a ‘Progress Record’ sheet (see Annex B) on which they documented their participation and perceived understanding. This record sheet identified the learning activities in which the student would typically need to participate to gain full benefit from the learning experience. Each of these activities appears as a ‘Yes-No’ question, such as ‘Have you answered the session questions?’ Using the ‘Progress Record’ sheet students were required to indicate, on a weekly basis, which of these activities they had undertaken. Students were also required to record a score between 0 and 5, representing their perception of the extent to which ‘they had learnt’ in that week.

Students could self-assess their understanding against the intended learning outcomes using a set of ‘Session Questions’. A score of 5 indicated that they had answered ‘Yes’ to all of the learning activities, had completed all the session questions, and ‘thought’ they had met all the session learning outcomes.

Answers to the ‘Session Questions’ (see Annex C) were provided on a regular basis (usually two weeks after the session to allow sufficient time for the students to attempt the questions, to promote review and reflection), allowing students to further assess and align their understanding. Students were required to compare their responses to the questions with these alternative answers, identify and record any differences and formulate any further questions to ask the tutor using an Activity Review Form (see Annex D). From this self-assessment students then recorded a score that could be compared with their original score.

During the workshop sessions the tutor monitored and reviewed with students the extent to which they were using the process and discussed any problems that had been recorded. The tutor maintained their own record of each of the student's scores (perceived and actual) and these were collected during the discussions. It was at this point that any difference between the student's recorded activities, score and the corresponding evidence was first identified.

In early workshop sessions the tutor monitored with each student their ‘Progress Record’ and viewed their portfolio evidence as necessary in order to promote the use of the process. In later sessions, however, when the tutor got to know the students, only random sampling of

their 'Progress Record' and evidence was required, unless students identified specific problems that needed to be addressed.

As a result of this process the students quickly became more proactive in raising issues that were more structured than the usual 'I don't understand' questions, in which students feel they don't understand but cannot identify the specific areas of difficulty. Further, by virtue of the process of formulating their own questions, the students had more ownership of the answers.

In designing the overall pattern of assessment one of a number of basic principles adopted was to ensure that if the students were unsuccessful it was not due to a lack of preparation for the type, style, format and expected standard of the assessment. For example, in order to prepare students for the end of unit examination, (some students had not taken the normal unseen examination in their previous courses), students were set examination style questions each week and later were provided with example structured answers as part of the learning programme. (See Annex E)

3 Issues

The approach took as a premise, perhaps most importantly, that any intervention initiated by the teacher to introduce any changes in the student's approach would only be effective if the student was inducted into the process. Therefore, initial activities were designed to aid this induction process. For example, the session questions in Annex 1.4.2 were designed to assess knowledge of the material introduced that week. Answers to the questions were either within the body of the session learning material or in the assigned further reading. If students did not have answers to the questions in the following week they had either not attempted the questions at all or had failed to undertake the further reading. Therefore, it was easy and quick to identify which students were not participating at least at the minimum level. The underlying reasons for this lack of participation were then discussed with these students.

A conventional solution to supporting failing learners is to allocate additional resources in the same form as those already provided. The key action of this type of approach focuses on the content of the learning and does not increase the opportunity for the type of dialogue necessary for learning. We recognised that in order to manage our workload providing additional resources alone was not especially effective. Rather, resources were more effective when they were targeted to the needs of the individual according to their level of engagement in the unit, as indicated by completion of the forms and evidence in their portfolio.

4 Benefits

A key strength of this approach lies in the fact that there are not insurmountable barriers to achievement. 'Slow starters' are accounted for, motivating them to learn by creating a link between effort and success. The approach provided a structured learning experience yielding better attendance, better student engagement and higher pass rates². It provided early identification of problems, by both teachers and students themselves and offered opportunities for positive and shaping feedback. Furthermore it provided the opportunities for students to develop good study habits. The two-way feedback facilitated by the 'Progress Record' sheet developed the student's ability to assess their own learning and in identifying

² These benefits have been seen over the last three years in this unit and claimed in other implementations of the generic framework by other tutors in a variety of units. This work will be published at a later date as additional case study material.

the nature of any problems. Most importantly, it was the student who initiated the feedback rather than the feedback coming as judgement from the teacher.

5 Evidence of success

Since this unit was introduced into the engineering programme as a 'broadening' subject it has not been perceived as being central to the students' specific engineering discipline in which they were interested. Therefore, they judged the unit to be of less relevance to them than the other units studied. This resulted in a lack of motivation, with students not completing early formative tasks designed to aid completion of the summative assessments later in the unit. This in turn led to poor pass rates both on an absolute level and as compared to other units in the programme. After the introduction of the 'Mexican Hat Approach', (see below) students engaged more in the learning process, which in turn led to an increase in the pass rate (75% in this case study), although they still perceived the unit as not being central. However, as a result of further developments of the unit's teaching and assessment strategy in line with the 'Mexican Hat Approach' model, the pass rate has now risen to 88% for the latest cohort, and this figure is expected to rise after September referrals.

The evaluation of data gathered during the operation of the unit in the 2003/4 session will be published at a conference later in the year (Robinson and Udall, 2004).

The framework has been presented to over 200 academics at various external and internal staff development events. We have also worked closely with other colleagues, at Southampton Institute and two other universities, in order to implement the approach in different contexts. All have seen greater engagement by the students in their studies.

Student feedback indicates that they would apply the framework in other units of study. Students particularly focussed on the way in which the structured activities and corresponding specific outcomes were made explicit and how these provided a way in which they could make judgements about their own performance.

6 The Mexican Hat Approach

The 'Mexican Hat Approach' (MHA) (Robinson and Udall 2003) underpins the framework presented in this case study. The MHA has also formed the basis of other learning, teaching and assessment strategies implemented in other subject disciplines and at other universities. It has been used as a simple technique using the framework to overlay existing practice, but still achieving the benefits of increased student engagement and better pass rates, within a manageable workload. However, for the full benefits to be realised the learning, teaching and assessment need to be aligned according to the educational theories and concepts of the model underlying the 'Mexican Hat Approach'.

The 'Mexican Hat' model focuses on the students being able to answer three key questions; 'How do I know if I am doing what I should be doing?', 'How do I know whether I have gained understanding from what I am doing?' and 'How do I know if the understanding I have gained will result in achieving success?'. It proposes three groups of students defined by the regions of the 'Mexican Hat'. These are, a) participating and achieving, b) participating but not yet achieving and c) not yet participating (and hence not yet achieving).

Those students who are participating and not yet achieving highlight the importance of formative assessment. This group of students, if not identified, is likely to perceive that they will be successful but are then surprised and disappointed (along with their tutors) with the results of the summative assessments.

The range of interventions associated with MHA are designed to allow students (and their teachers) to identify which region of the Mexican Hat they are in as the learning experience progresses. This facilitates the provision of appropriate support and guidance according to the needs of the students. As such it is a formative process but critically, driven by the student and not by the teacher.

7 Reflections

Before the introduction of this framework there were some students who had a perception of their performance that was not aligned with the required outcomes. These students appeared to believe that the end result would be positive, even though there was little foundation, in terms of engagement, in their studies upon which this belief could be based.

An important dimension of this framework is an attempt to link effort with success and align perception with reality, by making the activities that should be undertaken explicit and by having a recording method that allows students to recognise the extent to which they are engaging with these activities. The recording process itself diffuses any such self-delusion.

All of the activities associated with learning generated some evidence (ranging from answers to the session questions to work generated from attempting the workshop activities) that was maintained in their portfolio. When the teacher checked the 'Progress Record', they also used the evidence in the portfolio as a focus for conversations about the student's participation and attainment. Therefore, the framework made it difficult for students to hide from the fact that they were not engaging in the learning process sufficiently to achieve, as the evidence of their own work was always central to the teacher and learner interactions.

As a result of the conversations about attainment, students who undertook all of the required activities and who recorded a high self-assessment score were offered additional tasks that were designed to further broaden and deepen their understanding in the subject area. Those students who recorded low self-assessment scores or who had not completed all of the activities without help were offered additional complementary tasks, as appropriate, designed to improve their understanding and reinforce their learning. By following this process of diagnosis groups of students of widely differing experience or understanding could work at effectively their own pace.

8 References

Biggs, J, (1999) "Teaching for Quality Learning at University what the student does", The Society for Research into Higher Education, Open University Press, UK

Robinson, A and Udall, M (2003) "Developing the Independent Learner: The Mexican Hat Approach", *Conference Proceedings of the IEE 3rd International Symposium on Engineering Education*, Southampton, UK, January 6-7, 2003.

Robinson, A and Udall, M (2004) "Formative Assessment: Instigating Student Conversations about Learning", accepted for the *International Conference on Innovation, Good Practice and Research in Engineering Education*, University of Wolverhampton, UK, June 7-9, 2004.

Annex A – Structure of a typical session

Understanding the Software Problem

[Title of Session]

1.1 Session Outcomes

[There are specific and clear learning outcomes for each learning session]

At the end of this session you should be able to:

For example

- i. Identify some goals for software engineering
- etc. but not too many outcomes per session*

1.2 Body of Learning Material

[There is a body of learning material for each session. This is a mix of text, diagrams pulled together with a series of questions, the answers to which are discussed within the session.]

For example

Programmers often work at the computer and directly in the programming language defining the programme logic and defining programme components as and when required. Figure 1.2 shows the process a programmer uses to construct a source code program.

1.2a From the diagram, what are the two tasks that a programmer is performing simultaneously?

1.2b Assuming that the programmer has a basic idea of what the programme must do, why is there potentially a great deal of wasted effort in the programming process?

1.3 Session Summary

[A brief session summary is provided]

The Goals of Software Engineering

The approaches encompassed by Software Engineering set out to overcome the problems detailed in this session. The following could be considered some broad goals:

- i. To remove any wasted effort associated with programming;
 - ii. To develop ways of determining whether software is finished;
- etc.*

1.4 Post session Activities

[There are specific learning activities to complete before, during and after each session with criteria against which students are able to self-assess their progress.]

1.4.1 Further Reading

[In this case, one of the defined learning activities requires students to complete some further reading prior to the next session]

Chapter 1 of Sommerville (2001) provides an overview of the background to the software engineering discipline and introduces some of the key language used. Chapter 1 of Pressman (1997) offers an alternative perspective.

References

Pressman, R, (1997). *Software Engineering - A Practitioners Approach*. 4th ed. London: McGraw-Hill
Sommerville, I, (2001). *Software Engineering*. 6th ed. Harlow, Essex: Addison-Wesley

1.4.2 Session Questions

[In this case, there are a series of session questions that assess knowledge of the material (the only reason students may not have answers is that they didn't attempt the questions, as the answers are within the body of the session learning material or further reading.) This gives an easy and quick indication of whether the student is participating at least at a minimum level. There would also be more open-ended supplementary questions and problems later in the unit]

Once you have read the session learning material and finished the further reading, try these questions to test your understanding:

- i. What is meant by the term programme logic?
- ii. What are the two decisions that a programmer makes when programming?
- iii. Why could the programmer's model be considered to waste effort?
- iv. What is meant by the terms 'process model' and 'life cycle'?

1.4.3 Example Examination Question

[As there is an examination at the end of this unit, example examination questions are given each week in order to adequately prepare students for this type of assessment. For the early sessions a template answer structure is given to guide the student]

NB Remember that you should do the question under examination conditions i.e. in a set time, in a quiet place and without any course notes or textbooks to help you. In other words you should have done all the reading first.

1. Software may be produced iteratively by successively adding code until the product is 'finished'. At the end of this process the executable program (delivered to the customer) and associated source code are the only products.

- i) Explain why this approach may make it difficult to later find and cure errors found in the software by the customer [8 Marks]

Template answer structure.

Explain the reason for and effect of the following:

- Lack of structure in the code
- Not being able to see how the code works
- Not being able to relate the error to a specific bit of code
- Not knowing where and how to start testing

- ii) Suggest some ways of overcoming the difficulties encountered in part i. [12 Marks]

Template answer structure.

You may have had to do a lot of reading to find all of this answer. How might the following overcome the problems?

- Documentation to support the software
- Using some form of development life cycle

Pogress Record of Student Activity

[This form is used by the student to record their self-assessment against the learning outcomes. The Y/N answers indicate their participation, the score indicates their perceived understanding. This score is compared with a more objective score after the alternative answers are provided. Also the score is checked and recorded by the tutor thus providing feedback and a quick indication of the likely level of achievement. The evidence in the logbook is reviewed, on a random basis, or if there are doubts as to the students accuracy of self-assessment.]

Name

Unit Name

Course

Level

Note: Place a Y=Yes or N=No next to each learning activity to denote that it has been completed or not completed.

Learning Activity	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Lecture 1													
Activities 2													
3													
4													
5													
6													
Small Group													
Activities 8													
9													
10													
11													
Score													

Each week, next to score, rate between 0 and 5 how much YOU feel you have learnt for that week. A score of 5 means that all outcomes have been met and that you are satisfied that you have understood all of the material

A LOG BOOK SHOULD BE MAINTAINED AS EVIDENCE, RECORDING ANSWERS TO SET QUESTIONS AND ANY SOURCES OR REFERENCES USED

Learning Activities

1. Read the session material before attending the lecture session
2. Attended the lecture session
3. Reviewed the session material after the lecture session and determined whether the session outcomes have been met
4. Conducted the reading associated with the session
5. Answered the session questions
6. Conducted necessary preparation for small group session (workshop/tutorial/seminar)
7. Attended small group session
8. Completed work in small group session
9. Conducted further work from small group session
10. Read additional material to clarify understanding
11. Read additional material to extend learning

Annex C- Session Question Answers

[Students compare their answers with these alternative answers provided, record their more objective score (and compare it with their earlier perceived score), identify areas of uncertainty and formulate any questions for the tutor using the Activity Review Form]

Question Panel Session 1 – Answers

Once you have read the chapter and finished the further reading, try these questions to test your understanding

From the text

- i. What is meant by the term programme logic?

Programme logic is the expression of sequence, choice and repetition in software. Programme logic is associated with the detailed view of a software system.

(2 marks)

- ii. What are the two decisions that a programmer makes when programming?

The design decision of what order (sequence) the programme will have i.e. the definition of the programme logic itself.

How to express the programme logic in the programming language chosen

(2 marks)

- iii. Why could the programmer's model be considered to waste effort?

Fundamentally because there are lots of iterations of the "write code-compile-run" cycle. Changes made to the programme logic may introduce new errors and will inevitably mean syntax errors. For each pass of the outer loop (associated with the first decision above) there will potentially be many passes of the inner loop (associated with the second decision above).

(4 marks)

- iv. What is meant by the terms 'process model' and 'life cycle'?

A life cycle is a set of steps and stages that developers go through to make software.

A process model refers to a set of processes used to make software

Both of these are essentially the same.

(2 marks)

Annex D – Activity Review Form

[This form is used after the Session Questions and/or Example Examination Questions have been attempted, alternative answers provided and the student has compared their answers with these alternative answers, but still has areas of uncertainty they wish to raise]

Activity Review Form

Week Number:

Name:

Unit:

Course:

This box should be used to record any areas where further work is required, based on your comparison with the alternative answers provided.

This box should be used to record any questions for the tutor. You should have completed all learning activities, checked the alternative answers provided and made an attempt to find the answer from written sources first, before filling this in.

Score from the Progress Record of Student Learning (perceived score):

Score from comparison with alternative answers (objective score):

Annex E – Examination Question Answers

[Students compare their answers with these alternative answers provided, record their more objective score (and compare it with their earlier perceived score), identify areas of uncertainty and formulate any questions for the tutor using the Activity Review Form]

Examination Question Panel Session 1 - Answers

i. Explain why this approach may make it difficult to later find and cure errors found in the software by the customer

[8 Marks]

Explain the reason for and effect of the following

Lack of structure in the code

Code generated using the programmers' model evolves gradually as the system is developed. The decision to add new functions and new functionality is taken in an ad-hoc way and therefore the code becomes unstructured. This leads to the following problem ..

Not being able to see how the code works

Unstructured code is difficult to understand. Generally the design solution (how the system actually works) is hidden in the syntax of the language. Languages are often complex and difficult to understand and so looking at code on its own means that any errors are hidden. This is exacerbated by a lack of any documentation associated with the software system i.e. only the code exists.

Not being able to relate the error to a specific bit of code

*When an error appears in a software system it is necessary to associate the place where the error happens at run-time with the piece of code which performs the faulty operation. Finding the piece of code which performs a particular operation when only the code exists is hard to do. As with the previous problem, a lack of documentation means that knowing where to look is purely a code issue.
The programmers model tends to work with very informal requirements.*

Not knowing where and how to start testing

With the programmers model there is only a loose concept of what the system should do in the first place. This means that it is difficult to have a well-structured approach to testing either when the system is developed or when the system is operational. Without a clear idea of what a system must do it is impossible to know exactly what to test.

ii. Suggest some ways of overcoming the difficulties encountered in part i.

[12 Marks]

You may have had to have done a lot of reading to find all of this answer. How might the following overcome the problems?

Documentation to support the software

A clear statement of requirements in a requirements specification document provides a target for the systems development. By understanding what the system must do it is possible to devise structured tests to make sure that the software does these things correctly.

If the requirements themselves are well structured, it is also possible to relate each requirement to the specific bit of code which does that job. This is described as traceability. This allows the coded system to be tested against the requirements. In addition when an error appears the associated requirement can be found along with its corresponding code. This tells the developer where the faulty code is and this makes fixing the error easier.

Finally it is useful to have some representation of the software other than just the code. Models of the software can be used which clearly convey how it works. This improves the developers understanding of the 'internal workings' of the code and this in turn makes error detection and fixing easier.

Using some form of development life cycle

In order to manage the development of these documents, the overall software development process can be broken down into discrete phases with known activities and well specified inputs and outputs. Two key phases are associated with ease of error detection and fixing.

Firstly a design phase in which decisions relating to the structure and functionality of the software are made prior to coding. By making these decisions first, the system becomes more structured and this improved structure means that the system is easier to understand and hence errors are easier to find and cure.

Secondly, a proper structured testing process can be put in place during which specified functions in the software are tested against the design.

LTSN Engineering is the national centre for all engineering academics in the higher education sector who wish to enhance learning and teaching.

LTSN Engineering aims to promote quality learning and teaching in engineering across the UK by stimulating the sharing of good practice and innovation in learning and teaching through the provision of subject-based support. The Centre seeks to make a positive impact on current practice in learning and teaching through:

- Providing a national *focus* that is an accepted and essential point of contact for all involved in learning and teaching in higher education engineering.
- Collating and disseminating *good practice* and innovation in learning and teaching in higher education engineering.
- Providing co-ordination and *support* for the embedding of quality learning and teaching in higher education engineering.

Alternative Formats

This publication can be downloaded from the LTSN Engineering website, www.ltsneng.ac.uk.
Please call 01509 227170 for alternative format versions.

Contact us at:

LTSN Engineering

Loughborough University

Leicestershire

LE11 3TU

Tel: 01509 227170

Fax: 01509 227172

E-mail: enquiries@ltsneng.ac.uk

Discussion list: engineering@jiscmail.ac.uk

For more details about LTSN Engineering visit our web-site

www.ltsneng.ac.uk

engineering
