Australian Journal of Educational Technology

2003, 19(1), 46-58

Design principles for authoring dynamic, reusable learning objects

Tom Boyle
London Metropolitan University, UK

The aim of this paper is to delineate a coherent framework for the authoring of re-purposable learning objects. The approach is orthogonal to the considerable work into learning object metadata and packaging conducted by bodies such as IMS, ADL and the IEEE. The 'learning objects' and standardisation work has been driven largely by adding packaging and metadata to pre-constructed learning artefacts. This work is very valuable. The argument of this paper, however, is that these developments must be supplemented by significant changes in the creation of learning objects. The principal aim of this paper is to delineate authoring principles for reuse and repurposing. The principles are based on a synthesis of ideas from pedagogy and software engineering. These principles are outlined and illustrated from a case study in the area of learning to program in Java.

Introduction

Good eLearning resources are expensive to produce. The effectiveness of these resources, however, and the return on the investment made, has traditionally been limited by a number of factors. Resources developed within particular Virtual Learning Environments (VLE) or Managed Learning Environments (MLE), for example, could not be transferred for use in others. The challenge of making learning resources 'interoperable' across different systems thus became a major goal. At a higher level, tutors often wished to reuse and repurpose learning resources to meet the perceived needs of particular contexts and students. However, learning resources were often monolithic; the resources had to be taken on an all or nothing basis. The challenges of interoperability, reuse and repurposing of eLearning resources thus attracted considerable development effort (Duval, 2001).

The primary response to these problems has been the international work directed at developing learning object standards. The concept of learning object is defined very broadly. The IEEE standardisation draft defined learning objects as:

> a learning object is defined as any entity, digital or non-digital, that may be used for learning, education or training. IEEE (March, 2002).

The standardisation work has involved a number of major organisations, eg. IMS, ADL, and IEEE. The work proceeded in several parallel strands tackling different facets of the standardisation work. The two most significant strands pertinent to this paper are the work on metadata and learning object packaging. Metadata refers to the controlled taxonomy and related vocabulary used to describe learning objects. In June 2002 the IEEE agreed a standard for learning object metadata (LOM). This standard was based on a proposal developed by IMS (originally called 'Instructional Management System) which is turn had consolidated work from a number of other bodies.

IMS has also developed a proposed standard for learning object packaging. The basic proposal is to take any learning object and provide a 'wrapper' around this object. This wrapper describes the component structure of the object, and includes the descriptive metadata. The learning object is thus 'packaged' in a standard container format. This packaged object can be stored in digital repositories. The metadata permits fast effective searches to retrieve learning objects suitable for a particular purpose (e.g. Koppi & Hodgson, 2001). These learning packages should then be interoperable across different LMS (learning Management Systems) as the vendors bring their tools into compliance with the standards. The SCORM reference model provides a higher level framework which relates these major strands within the broader work on standards development (SCORM, 2002; RHA Associates, 2002)

This work is very valuable. It is making a very significant impact on the evolution of eLearning. Calverley (2002) provides a good guide to the relevance of this work to creating re-usable learning materials. The central argument of this paper, however, is that this approach is not enough. In order to provide a non-contentious basis for standardisation, a learning object is defined to be almost anything. The standards are declared to be pedagogical neutral (IEEE, 2002). The approach thus does not make any statement about the authoring of learning objects. However, there is a marked limit to the productive reuse and repurposing of learning objects that have not been designed for these purposes in the first place. There is, in the end, a limit to what can be achieved by intervention after the event (after the design and authoring process). We cannot, of course, change the past. In the future, however, learning objects must be developed with potential reuse, and especially repurposing in mind. The principal aim of this paper is to explore and delineate principles underlying authoring for reuse and repurposing.

Towards a synthesis of software engineering and pedagogical principles

Software engineering is concerned with the design, development and maintenance of large complex software systems. A major challenge in software engineering has been the issue of developing 'maintainable' systems. The use of the word 'maintenance' here underplays the nature and scale of the problem. Software systems evolve over time to meet the developing needs of the commercial context in which they are used. The software thus has to be adapted to meet these new challenges. It is claimed that over 70% of commercial software engineers' time is spent on 'maintenance'. However, changing software is a difficult and error prone process. The discipline of software engineering has thus developed principles for the development of systems that are designed to be 'maintained'. A principal focus for several of these principles is appropriate modularisation - breaking the whole into software units designed to ease the maintenance problem. These principles have direct relevance to the development of learning objects that are designed for re-use and repurposing.

The first principle in that of *cohesion* - each unit should do one thing and only one thing (Sommerville, 2000; Pressman & Ince, 2000). A direct link can be made to the idea of learning objectives in pedagogical theory. This mapping suggests that each learning object should be based on one learning objective or clear learning goal. This may be illustrated by the work on introductory programming in Java referred to later in the paper. There are, for example, three types of loops (language constructs for repeating blocks of code) in Java. Textbooks usually treat these together. The principle of cohesion, however, indicates that there should be a separate learning object for each type of loop. An immediate advantage is that the tutor can select the order in which these learning objects are combined. A tutor dealing with experienced student may wish to deal with these in sequence; another tutor with a different group of students may intersperse these learning objects with object dealing with other features of the language.

In order to provide this freedom to order learning objects a further design principle is important. This is the principle of 'de-coupling', or more accurately minimised coupling. This principle states that the unit (software module/learning object) should have minimal bindings to other units. Thus the content of one learning object should not refer to and use material in another learning object in such a way as to create necessary dependencies. One object then cannot be used independently of the other (Sommerville, 2000; Pressman & Ince, 2000).

This principle is crucial in design for reuse. The learning object should, as far as possible, be free standing. For example, a learning object on one type of programming loop should not refer specifically to content covered in another object. If we move the object it should still be fully understandable and function to achieve its learning objective. The vision then is of a group of cohesive and decoupled learning objects that can be selected and ordered to provide different learning experiences. This provides one type of adaptation based on inter-object selection and ordering.

The decoupling of learning objects is a considerable challenge. As eLearning designers we tend to think of the overall impact on learning, and strive to achieve rich, integrated learning experience. The challenge is to maintain this richness in a system composed of reusable components. There are a number of significant advantages, however, in taking on this challenge. These advantages are explored in the example on learning objects for introductory programming described later in this paper.

There is a final, crucial challenge that must be tackled to make these learning objects rich and effective learning experiences. It would certainly be possible to create a list of learning objects that are cohesive and relatively decoupled, but are also pedagogically barren. We must face the challenge of creating learning objects that are cohesive, decoupled and pedagogically rich. This design challenge is associated with the issue of 'repurposability' as we might expect rich learning objects to provide further options for adaptation by local tutors.

Rather than pursue the argument at a more abstract level, it is useful at this stage to study the realisation of these principles in a concrete implementation. This study concerns the developments of learning objects for introductory programming in the Java language. It will be used to illustrate how the challenges of cohesion, pedagogical richness and decoupling are being tackled. This will then provide the basis for a further clarification, in the later part of the paper, of the principles involved.

Learning objects for introductory programming in Java

Java has become a very popular candidate for the teaching of introductory programming at university level. Java meets the constructivist criterion of being an 'authentic' topic for study (Grabinger & Dunlop, 1995). It is a powerful, real world language that can be used to create applets for the Web or full software systems. Tutors also like the language because it embodies the object oriented paradigm that is so influential in modern computing. There are thus good reasons for teaching Java. Unfortunately, many students find it difficult to learn. Even universities that can select

from among the best students report difficulties in teaching Java. Thus Jenkins and Davey (2001) state:

> Anyone who has presented an introductory programming module will be all too familiar with students who appear to be totally unable to grasp the basic concepts. Others who come to supervise final year dissertations will have been faced with students who insist that they want to avoid programming at all costs.

To tackle this problem the School of Informatics and Multimedia Technology at the University of North London instituted a project beginning in Spring 2002 to substantially improve the learning experience for first year students of programming. In August this became part of the Department of Computing in the new merged institution, London Metropolitan University. The study involves a large group of over 600 students. The project involves intervention in syllabus development, the social organisation of learning and the introduction of new eLearning materials. The eLearning resources are being based on the authoring of rich, reusable learning objects. This development provides the focus for the present discussion.

The university is a partner site in the UK LTSN National Subject Centre for the Information and Computing Sciences. This Centre is funded by the four UK national higher education funding councils to provide advice and support in teaching and learning to all higher education departments in the UK in Computing and Library and Information Science (LTSN-ICS 2002). The present project is acting as the preliminary step in exploring the potential of setting up a national repository of learning objects for introductory programming. The learning objects are being developed both to meet immediate pedagogical needs and to serve this larger goal. This produces extra pressure initially. However, it provides the potential to divide the eventual task among a number of contributing partners, exploiting considerable advantages of scale.

This project is dealing with a real and urgent problem. The initial set of learning objects had to be developed and used within a tight time frame. It was planned that refinements to the learning object structure could be implemented on the basis of feedback from real evaluation data.

The main design requirements for learning objects may be summarised as follows. Each learning object should be based on one clear learning goal. From a software engineering perspective each learning object should be as cohesive and de-coupled as possible. This greatly facilitates re-use and re-purposing. From a pedagogical perspective, however, there is the need to create an overall coherent learning experience. These design challenges

may be in conflict. A key challenge for the project is to resolve the tensions in a creative and productive way.

Compound objects

The software engineering principles imply that learning objects should be as simple as possible. This greatly aids recombination and reuse. However, such simple objects may well appear pedagogically unexciting. Swan (1994), for example, argues that providing multiple perspectives aids learning. The multimedia resources available for the Web certainly enable the creation of rich, alternative ways of viewing and traversing a given learning topic. How can the use of these powerful techniques be squared with 'simple' learning objects?

One solution adopted is the creation of compound learning objects. In language a compound sentence is a sentence that consists of several independent clauses – 'I went to New Zealand and I attended ASCILITE'. Each clause can stand on its own as an independent entity. (These sentences may be contrasted with complex sentences which contain bound or dependent clauses - 'I went to New Zealand because …').

A learning object may be thus simple, consisting of one independent object, or it may be compound. A compound object consists of two or more independent learning objects that are linked to create the compound. There are two main advantages of compound objects:

1. They provide pedagogical richness not available through simple objects.
2. They provide a significant basis for re-purposing.

A further important feature is that each simple component object can be reused independently.

Compound objects support alternative views of the same learning issue, eg. as a text based explanation or as a multimedia animation. They thus provide a basis for pedagogical richness that fully exploits the opportunities offered by the technology. It provides a basis for repurposing through the addition or deletion of objects to amplify or shape the pedagogical richness of the compound object. Local tutors may be presented with a default compound, but they should be able to reconfigure this to shape their own compound object.

This concept is being implemented in the Introductory Programming project at London Metropolitan University. These learning objects are being developed to meet an urgent practical need. The structure

developed thus represents one presentation format for compound learning objects. This presentation format treats the textual explanation, expressed succinctly on a Web page, as the basic entry point into the compound.

Example of a compound object

Computer languages can be decomposed into basic building blocks. Each building block may be associated with solving a recurrent problem in writing computer programs. The 'learning objects' are based on these basic components. Each compound object consists of a web page consisting of two main parts. The first part is a succinct textual explanation (Figure 1). This can operate as an independent learning object. The second part is the 'link' column. This provides links to other objects (often multimedia objects) that amplify the learning experience offered by the compound object. Each of these linked objects is structured so that it can be used independently of the text based object. This is laid out schematically in Figure 1.

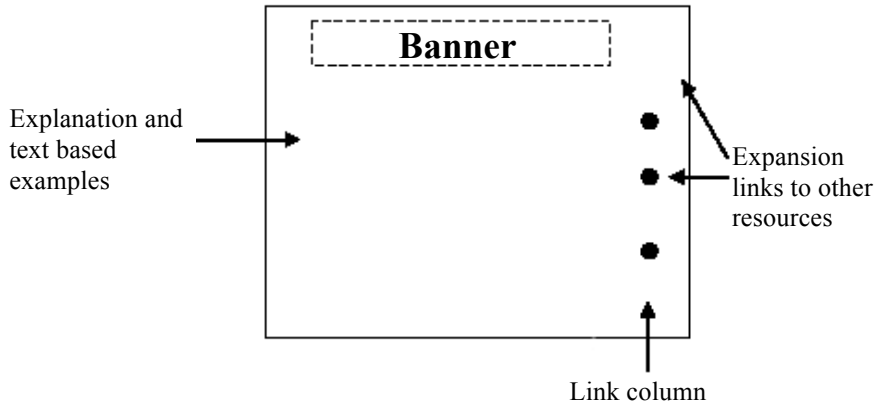

Figure 1: Schematic layout of format for learning object realisation

The structure of this layout is very simple but also very flexible. It implements a basic design pattern for multimedia (Lyardet, Ross & Scwabe, 1998). The purpose of this design pattern is to manage the bindings between one object and others. If we are to have cohesive and relatively decoupled learning objects then we must have a design mechanism for managing these bindings. There are two main types of binding: navigational bindings through URLs and non-URL based content bindings. This design pattern deals with the issue of URL based bindings.

The primary design feature is that the URLs must not be mixed in with content. They must be kept and managed on a distinct area of the screen.

This produces minimal explicit bindings between the main content and the URL links. The URLs can be added to, subtracted from, or modified without affecting the core object structure.

This provides an important mechanism for 'repurposing'. A learning object consists of a core and zero or more expansions. A default object is presented with the core with certain expansions added. These expansions aim to provide added pedagogical value to help in attaining the learning objective. However, as the couplings are precise, locatable and minimised, it provides a basis for changing the objects to meet specialist or evolving needs. These objects can be repurposed through the addition, subtraction or modification of extra resources. This approach provides a basis for the development of rich, adaptable learning objects through the management of the coupling relationship within a compound learning object.

Illustration of an adaptable, compound learning object

Appendix 1 provides an illustration of a learning object developed using this format. The learning objective is to enable students to comprehend, and use in simple programs, the Java code for instantiating objects from classes (this is the basic technique for the reuse of software in Java). The core of the object contains a succinct text description providing example code and an explanation of the Java constructs. It aims to do this in language appropriate to a learner, and thus introduces the technical terms in a 'Jargon' section at the end.

This object has a number of expansions (Appendix 1). There is no compulsion on a student to use these. A student who has experience of other programming languages may find that this textual explanation suffices. A novice student may prefer to work through all the expansions available. One of the expansions provides a Java applet that provides sample code in a full applet and executes the sample code. This is accessed through the expansion point labeled 'run applet'. A further resource is provided by a Flash based interactive movie that gives an animated illustration of the instantiation of an object. This resource is accessed through the slot 'run animated explanation' (Appendix 1). A screen dump from this animation is given in Figure 2. The animation culminates in a simple game where the student can select individual 'words' and construct the appropriate Java code. The aim of this resource is to provide an attractive resource that amplifies the pedagogical richness of the learning object.

The animated resource, of course, is a learning object in its own right. It is self contained. So although it is used to provide an enriched extension to the text object here, it is not bound. It could be used on its own, in a

lecture, for example to illustrate the underlying concepts. It is important for reusability that the resources also act as independent reusable objects. The fact that it is an independent object also has advantages at the authoring stage. The development of the text object and 'expansion' objects can proceed in parallel.

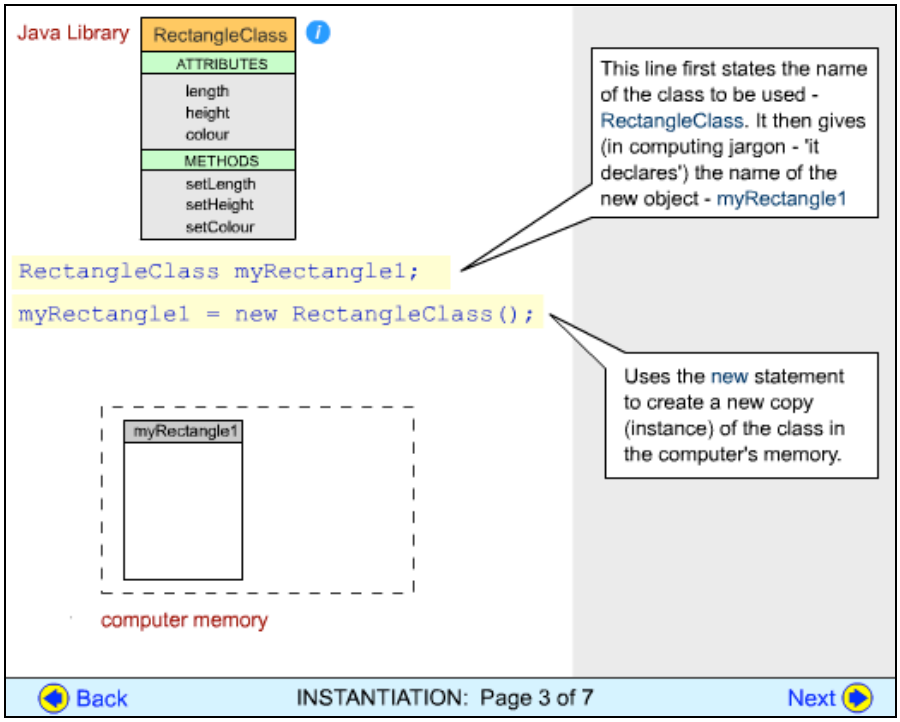

Figure 2: Screenshot from the animated explanation of instantiating objects

Learning objects and course structure

There is a further, and more obvious, dimension to decoupling. This concerns the relationship between learning objects and the syllabus, course or other higher organising structure in which they are delivered. Learning objects should not be coupled / bound into particular course structures. In terms of Web based implementation, this means that the syllabus navigation structure operates at a different layer of organisation for the learning object resources (which can be reused in different syllabus structures). The 'syllabus' navigation panel should be held as a separate object (Figure 3). The syllabus can thus be re-purposed easily by the addition, subtraction or re-ordering of links in the main

syllabus/navigation 'menu'. The only link from the syllabus to a particular learning object should be one URL. The learning objects are thus as decoupled from a particular syllabus as possible. The local tutor should thus be able to repurpose the syllabus and/or the learning objects.

Many different syllabi may be created to meet different needs, eg. university courses or short courses for industry. These syllabi objects operate at a different layer from that of main content objects (Boyle 2001, Boyle & Cook, 2001). The layers thus provide different levels of organisation, and the links between objects at different layers should be as clear and controlled as possible.

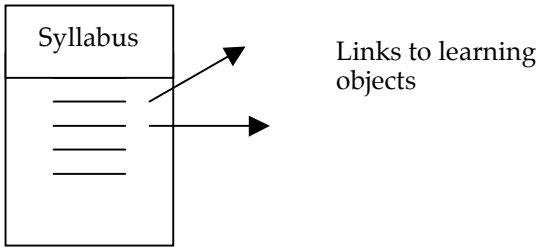

Figure 3: Schematic representation of a syllabus structure

The discussion of this topic has been considerably simplified as it is not the primary focus of this paper. The key message is the need to establish distinct layers of organisation in eLearning. The relationship between entities at different layers should be managed in an explicit manner that emphasises the principle of decoupling.

Ongoing development work and futures

The paper has set out a series of design principles for the design and authoring of learning objects. The central challenge is to design for reuse and repurposing. These principles have been illustrated with learning objects developed for a project to improve the learning of Java. This project is addressing a number of ongoing challenges in achieving maximised decoupling of the learning objects.

These eLearning resources are being used with a cohort of over 600 studying Introductory Programming in the period September 2002-Janaury 2003. It is not enough that these objects satisfy formal criteria of cohesion and decoupling; they must also be effective pedagogically. A Research Fellow has been appointed to carry out a detailed evaluation of

the impact on learning and student acceptance. It is intended that this evaluation should provide information directly on the pedagogic value of individual learning objects. The preliminary results are positive and encouraging.

A further stage of development is to use these quality assured objects as the 'seedcorn' for a national repository of learning objects managed through the UK LTSN National Subject Centre for Information and Computer Sciences. This initiative would support the parallel development and exchange of learning objects at different higher education centres. The full advantages of cohesive, reusable learning objects can only be achieved by creating communities that develop and exchange learning objects.

Acknowledgements

The author would like to acknowledge the contribution of Richard Haynes from the Teaching and Learning Technology Centre, London Metropolitan University, who carried out the Flash authoring of the illustration provided in Figure 2.

References

Boyle, T. (2001). Towards a theoretical base for educational multimedia design. *Journal of Interactive Media in Education*. http://www-jime.open.ac.uk/2001/boyle/boyle.html

Boyle, T. & Cook, J. (2001). Towards a pedagogically sound basis for learning object portability and re-use. In G. Kennedy, M. Keppell, C. McNaught & T. Petrovic (Eds.), *Meeting at the Crossroads*. Proceedings of the 18th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education. (pp. 101-109). Melbourne: Biomedical Multimedia Unit, The University of Melbourne. http://www.ascilite.org.au/conferences/melbourne01/pdf/papers/boylet.pdf

Calverley, G. (2002). Distributed Learning Project Guide: Creating Reusable Materials. [viewed 1 Oct 2002, verified Word doc 29 Jan 2003] http://www.cetis.ac.uk/groups/20010809144711/FR20020618103339

Duval, E. (2001). Standardized metadata for education: A status report. In Montgomerie, C. and Jarmo, V. (Eds), *Ed-Media 2001, World Conference on Educational Multimedia and Hypermedia*. AACE, pp. 458-463.

Grabinger, R. S. & Dunlop, J. C. (1995). Rich environments for active learning: A definition. *ALT-J*, 3(2), 5-34.

Appendix 1: Example - Creating software objects from class templates

Creating software objects from class templates

Problem

In Object Oriented programming we need to create **objects** from **class** templates. How is this done in Java?

Example code

```
RectangleClass          myRectangleObject;
myRectangleObject = new RectangleClass ( );
```

Run applet

Explanation

In Java creating a new object is achieved in two steps:

Step 1: give the object a name and indicate which class it belongs to as follows:

```
RectangleClass      myRectangleObject;
                           ↑
   class name       object name
```

This line first states the name of the class to be used - RectangleClass. It then gives (in computing jargon - 'it declares') the name of the new object - myRectangleObject.

Step 2: use the **new** statement to create a new copy (instance) of the class

Run animated explanation

```
myRectangleObject  =  new RectangleClass;
                      ↑          ↑
   object name      command    class
```

This can be read as create myRectangleObject as a new object of the class RectangleClass. This command produces one instance (copy) of the class in the computer's memory. We can now manipulate that software object (e.g. change the size, colour or position of the object)

Jargon

When we give the name of object - we **declare** the name of the object. When we create a new object from a class template - we **instantiate** the class (i.e. create an instance of the class)

Jenkins, T. & Davy, J. (2001). Diversity and motivation in introductory programming. *Italics*, 1(1). [viewed 1 Oct 2002, verified 29 Jan 2003] http://www.ics.ltsn.ac.uk/pub/italics/issue1/tjenkins/003.html

Koppi, T. & Hodgson, L. (2001). Universitas 21 learning resource catalogue using IMS metadata and a new classification of learning objects. In Montgomerie, C. and Jarmo, V. (Eds.) *EdMedia 2001, World Conference on Educational Multimedia and Hypermedia*. AACE, pp.998-1001.

IEEE (2002). Draft Standard for Learning Object Metadata. [viewed 4 Mar 2002, verified 29 Jan 2003] http://ltsc.ieee.org/doc/wg12/LOM_WD6_4.pdf

LTSN-ICS (2002). The LTSC-ICS Website. [viewed 1 Oct 2002, verified 29 Jan 2003] http://www.ics.ltsn.ac.uk/

Lyardet, F., Ross, G. & Scwabe, D. (1998). Using design patterns in educational multimedia applications. In T. Ottmann and I. Tomek (Eds), *EdMedia and Ed Telecom '98. Procs. of the 10th World Conference on Educational Multimedia and Hypermedia*. AACE.

Pressman, R. S. & Ince, D. (2000). Software engineering: A practitioner's approach. 5th ed. - European edition. McGraw-Hill.

RHA Associates (2002). SCORM overview. [viewed 1 Oct 2002, verified 29 Jan 2003] http://www.rhassociates.com/scorm.htm

SCORM (2002). ADL Website. [viewed 1 Oct 2002] http://www.adlnet.org/

Sommerville, I. (2000). Software engineering, 6th Ed. Addison-Wesley.

Swan, K. (1994). History, hypermedia and criss-crossed conceptual landscapes. *Journal of Educational Multimedia and Hypermedia*, 3(2), 120-139.

This article was nominated for an Outstanding Paper Award at ASCILITE 2002, gaining the additional recognition of publication in AJET (with minor revisions). The reference for the Conference version is:

Boyle, T. (2002). Design principles for authoring dynamic, reusable learning objects. In A. Williamson, C. Gunn, A. Young and T. Clear (Eds), *Winds of Change in the Sea of Learning: Proceedings of the 19th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education,* pp57-64. Auckland, New Zealand: UNITEC Institute of Technology.

Author: Tom Boyle
Learning Technology Research Institute (LTRI)
London Metropolitan University, United Kingdom
t.boyle@londonmet.ac.uk