Learning Object Patterns for Programming

Ray Jones and Tom Boyle
Learning Technology Institute, London Metropolitan University, London, UK

r.jones@londonmet.ac.uk t.boyle@londonmet.ac.uk

Abstract

This aim of this paper is to show how existing learning objects, that have previously proved to be successful, can be used to derive patterns that could be reused in the design of new learning objects. This is demonstrated in the context of learning objects that were designed to aid the mastery of computer programming by learners who had had no previous experience in the subject.

Learning computer programming presents significant challenges to newcomers to computing so the reuse of successful learning object design has the potential for real pedagogical benefits.

Patterns are a well-known design technique in the fields of architecture and software engineering. In a similar way to their use in object-oriented software design, patterns for the design of learning objects can be derived from successful existing learning resources; these patterns can then be reused in the design of new ones.

This paper describes the learning objects that were designed to aid new computer programmers and how patterns were extracted from those learning objects. This results in a small learning object pattern catalogue that has the potential for reuse in the construction of new learning objects.

Keywords: Learning objects, reuse, patterns, templates, design, authoring.

Introduction

The aim of this paper is to demonstrate how design patterns can be derived from existing learning objects and in order that they may be reused in the design of new ones. This can give tremendous benefit in the design of new resources as they can be based upon those that have already proved to be effective. Reusing effective design like this is beneficial in two ways: first, the new resources can inherit the effectiveness of the originals; and second it will be quicker and easier to create the new resources from a pattern rather than starting from scratch.

By attending to appropriate criteria for reuse and ensuring that the patterns and templates incorporate these features, the learning resources derived from the patterns will also be reusable within contexts similar to those for which they have been designed.

As Goodyear (2005) states, there is a large unmet demand for guidance in design, in particular, where it is given in the form of reusable and customisable ideas. It is intended that the implementation of design patterns as reusable templates will contribute to satisfying this demand.

Editor: Alex Koohang

The remainder of this paper is structured in three parts.

In the first section of the paper, the background to the work is described in terms of the resources that will be used for demonstration purposes, the criteria for reuse of learning resources and design patterns.

The second section is concerned with the analysis of the resources (learning objects) and the initial identification of patterns within them.

The third section of the paper describes the patterns more formally as a small design patterns catalogue.

Background

The Learning Objects

The resources used for this work are well-used and proven learning objects that were designed to complement a first-year degree course in programming. In this course, the learning objects proved to be a valuable contribution to a significant enhancement of student success and a major improvement in pass rates for the course. Their successful use is well documented as part of an innovative blended learning approach to the teaching of programming to first-year computing students (Boyle, Bradley, Chalk, Jones, & Pickard, 2003). The learning objects won a European Academic Software Award in 2004 (EASA, 2004).

The learning objects were designed by a team of subject experts and a multimedia developer who authored them in Flash. Each learning object was self-contained and was made up of a sequence of frames that, themselves, consisted of smaller, self-contained, multimedia content. The aim of each learning object was to explain a particular programming concept in the Java programming language through the use of examples. (Examples of these learning objects can be found at http://www.londonmet.ac.uk/ltri/learningobjects/examples.htm.)

The main learning objects, their content and the sequencing of that content, were designed to implement a particular pedagogical method that is described in the next section.

Pedagogical Method

At the simplest level the structure of the learning objects is in three parts each of which is implemented through one or more multimedia 'pages'. The first part is an introduction, where the learner is introduced to a new topic; the second is the main part of the learning object where the learner will gain an understanding of a new topic; and in the last part the learner applies their new understanding.

The first part of the learning object is implemented by a single page containing a simple description of the topic to be engaged with.

The second, understanding, phase of the learning object can be expanded into three sub-phases: Engage, Apprehend and Comprehend.

The Engage sub-phase is realized through a page with two components – a commentary linked to a familiar, visual example of the new ‘abstract’ concept. The aim here is to gain the learners attention by providing a familiar and easily understood example of the concept that is about to be tackled.

The Apprehend sub-phase introduces the target code construct as a third component. For our purposes 'apprehension' can be thought of as the point where the learner gains a holistic view of the concept that is being introduced. The aim of this sub-phase is to see (apprehend) the overall effect of the code before going into details. An animation is used to illustrate the overall effect of executing the code.

Comprehension is where the new concept is engaged with at a more logical level. This is typified by a step-by-step analysis of the new concept or idea. The Comprehend sub-phase is realized through a frame that provides an animated step-through of the program code illustrating the effect of each step.

The three components are synchronized through the use of buttons embedded within the textual commentary.

The learning object finishes with a scaffolded construction exercise where learners can, under supportive conditions, actively create a program fragment that illustrates the concept being learnt. This facilitates the transfer to 'real life' conditions where the learner will need to use a standard programming editor/compiler to construct and test the code.

It is interesting to note that, unlike other approaches, the last part of the learning object is not viewed as a closure test; rather, it is an active engagement that supports the transition to applying the new understanding in a real life situation.

While this method relates to Kolb's *Experiential Learning* (1985), it was developed empirically and, as stated above, has been shown to be an effective contribution to improving the success of students in learning programming skills.

It should be noted that while, above, we provide a simplistic explanation of apprehension and comprehension that is sufficient for the purposes of this paper, a more detailed explanation of these ideas is given in Kolb (1985).

Learning Object Reuse

The arguments in favour of reuse of learning resources include economic ones (Downes, 2001) and those of quality (Jones, 2004): economically, the reuse of resources rather than their re-invention must save both time and effort, and therefore money; and, from a quality point of view, the more a resource is reused the more likely it is to be of a high quality simply because more people will have been exposed to it and will have had the opportunity to provide feedback on it.

However, in order for learning resources to be effectively reused they should be designed with reuse in mind: they should be cohesive and decoupled from other resources (Boyle, 2003). In other words a reusable learning object should be concerned with a single topic (i.e. cohesive) and not be unnecessarily linked to (i.e. decoupled from) external resources, thus a learning object should be "an independent and self-standing unit of learning content" (Polsani, 2003).

Design Reuse

There is another type of reuse, other than simply reusing content. That is the reuse of design. It is this type of reuse that is a feature of object-oriented programming (OOP) and design. In OOP, objects consist of two parts: data and operations that may be made on that data. For example, in a computer program that is concerned with staff employment, there may be objects that represent individual employees. Such objects would contain data such as the employee's name, their salary and other personal details such as the number of hours worked. The operations that the object might contain could be giving the employee a pay rise, or calculating a monthly salary. The objects that represent each of the individual employees of a company contain the same type of data and the same operations; what changes is the specific value of the data for each employee. Thus the design of each of the employee objects is the same.

In OOP the design of an object is defined in a *class*. A class is like a template that defines the individual data that an object will contain (but not its value) and the operations that may be made on the data. In an object-oriented program the individual objects may be automatically created, as required, from the previously defined class. This is the most common type of reuse found in OOP: it is not the reuse of content but rather reuse of a design, i.e. the class is used in a program, as many times as is required, to create new objects. Furthermore, once a class has been defined, it can be stored in a repository and reused in a different computer program.

Applying this type of reuse to learning objects would allow a successful design to be used to produce new learning objects that shared the same structure and methods as the original ones. This structure could, and should of course, include the criteria for content reuse described in the previous section.

Another method of object-oriented design is the use of Design Patterns. Patterns are similar in function to classes but their scope is larger, so that a design pattern may incorporate many classes. Patterns, again, are a method of design reuse.

Patterns

Expert practitioners use their experience of solving problems in the past to build on and create new solutions in new situations. "One thing expert designers know not to do is to solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past. When they find a good solution they use it again and again. Such experience is part of what makes them experts" (Gamma, Helm, Johnson, & Vlissides, 1995). These reusable solutions that Gamma et al. refer to are known as design patterns.

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Alexander, Ishikawa, & Silverstein, 1977). Here, Alexander et al. were referring to patterns in architecture and town planning but core solutions in any area of design can be described in terms of patterns.

It is these patterns that will be extracted from the learning objects referred to above for reuse in creating new learning resources. The aim in doing this is to help learning object designers by providing them with a set of design ideas in a structured way that will clearly articulate the design problem and its solution (Goodyear, 2005). This will help enable the designer to produce effective learning resources in an efficient manner.

Analysis of the Existing Resources

As mentioned above the learning objects from the Java project implemented a particular pedagogical method that resulted in a simple sequence of 'pages' of multimedia content. Each of the pages was of a particular type, or design, and served a particular purpose. Each learning object comprised a similar sequence of pages. User control over the sequence consists of simple *forward* (go to the next page), *back* (go to the previous page) and *rewind* (go to the first page) buttons.

In a typical learning object, the first page was a simple HTML page that provided the title and a short description of the topic to be covered. This was followed by pages of different types from simple static text and images to pages that consisted of synchronised sequences of animations and text.

The general format of each sequence, and a description of each page in the sequence, is given below.

1. The title page

The title page gives the name and a brief description of the learning object. There is also a *start* link on the page to provide a method to begin the sequence. This represents the first part of the pedagogical process described above.

2. The concept page

On this page, the aim is to introduce a concept and explain it in a context that the learner is already familiar with.

The page comprises two vertical windows: in the right window is a textual description of the concept and in the left window is an animation that illustrates the concept. There is a button in the text window that starts the animation.

This page implements the first phase of the 'engage' part of the pedagogical process.

3. The example page

Next the concept is demonstrated in the context of the Java programming language. The page comprises three windows: one contains the Java program; the second holds a graphical image that will illustrate the operation of the Java program; and the third is a text window that has an explanation of the program. There is a button within the text window that starts the animation.

This page implements the second phase of engagement: it is here that the learner 'apprehends' the problem.

4. The detailed explanation page

Here the same example is given as in the previous page but in this case the program code is stepped through one line (or a small section) at a time, in the order of program execution.

Again there is a button in the text window that starts the animation but this time all windows are animated. The animation illustrates the code being executed by highlighting the appropriate parts of the code that are being executed. At the same time the animation is also stepped through and the text window changes at each step to describe the action of the particular line, or section, of the program code.

Thus there are three parts of the page that are cycled through in synchronisation: the code, the commentary and the animation.

This is the final part of the engagement process where the aim is comprehension.

5. The practice page

This final page poses a problem for the learner. In order to solve the problem the learner must select fragments of program code in order to construct a valid program or program fragment. The page is constructed in three parts: at the top are the textual instructions for the learner; at the bottom are the lines of program code that need to be selected; and in the middle is an initially blank area where the resulting program will be displayed.

To construct the program the learner simply clicks on the correct lines of code in order. These are then copied into the centre area of the page where the complete program fragment is constructed. The learner is given feedback on the choices that are made so that any mistakes may be corrected.

This page is the scaffolded exercise referred to in the pedagogical method.

It can be seen from this that there were five basic page types. A typical sequence consisted of at least five steps as described above. However, in some learning objects steps 3 and 4 were repeated with different programming examples and, often, there was more than one practice page. Thus a typical learning object may consist of a sequence of more than five steps.

Learning Objects as Patterns

A major aim of this work is to show that design patterns can be derived from existing, successful, learning objects, in order that they can be reused in the design of new learning objects. These new resources may incorporate new or existing materials and cover different topics to the original.

From the analysis, above, a pattern can be detected in the sequence of pages that make up each learning object. The individual page types can also be regarded as patterns that describe a particular learner activity.

Thus the main pattern may be considered as a simple sequencer of pages that introduce a new programming concept. This pattern can be regarded as a compound pattern (i.e. one that is made up of other patterns).

Buschmann, Meunier, Rohnert, Sommerlad, and Stal (1996) define design patterns, in part, as describing a particular recurring design problem that arises in particular design contexts and presents a well-proven scheme for its solution. The learning object components, described here, have been extracted from existing successful learning resources and they are described in a technology neutral manner, thus they can be considered as design patterns. However, Buschmann et al. also suggest that patterns should be described in a consistent and structured manner in terms of their context (the situation that gives rise to the problem), the problem itself and the solution to that problem. This advice follows the lead of Alexander et al. (1977) who describe their patterns in a similar manner and who refer to a collection of such patterns as a pattern language.

It is also important that the learning object components that are defined using these patterns fully take into account the need for reuse and adaptability. Thus these aspects need to be addressed in any documentation of a pattern.

Thus the patterns, so far identified, should be documented as described above.

A "Learning Objects for Programming" Pattern Catalogue

The following descriptions form a small patterns catalogue for learning computer programming concepts.

New programming concept

Context: Learners often find computer programming difficult. Failure to grasp these concepts impedes their progress in learning to program.

Problem: In order to learn a new programming concept and be able to apply this concept in a program, it is valuable to start from a position that the learner is already familiar with. Once comfortable with the concept, examples may be given in the programming language of choice. Such examples need to be explained in a step-by-step manner in order that the learner can begin to apply the new concept in their own programs. Finally, the learner needs to be able to practice writing programs of their own but in order to be confident in doing this it is helpful to provide a simple environment where the learner can practice their new knowledge before being faced with the complexities of a real programming environment.

Solution: Use a sequence of multimedia pages to implement each of the learning stages described above.

Title page

Context: The learner is about to engage with a learning resource that he or she has not encountered before.

Problem: It is useful for the learner to have advanced warning of what is to come in order that they may prepare themselves and be given instructions on how to use the resource.

Solution: The title page provides a space for a title and a description of the resource.

Concept page

Context: the learner will be presented with a new concept in programming.

Problem: Programming concepts are abstract and are sometimes difficult for learners to grasp. Introducing a similar concept from everyday life, or from a topic the learner is known to be familiar with provides help in introducing the new concept.

Solution: Provide a resource that describes the familiar concept and then relates it to the programming concept.

Example page

Context: The learner is now familiar with the concept to be covered but is not familiar with its particular implementation in a programming language.

Problem: The learner is unaware of how the new concept is used in the language being learnt.

Solution: An example of the implementation of the concept in the language being learnt is provided along with a commentary and an animation that shows what happens when the example program code is executed.

Detailed explanation page

Context: When a new programming concept is to be learned it is valuable to be able to see an example program, or program fragment, and to have its operation explained in an interactive way.

Problem: In order to properly illustrate a programming concept or structure it is valuable to 'walk through' an example. In a teacher-led class this is easily achieved but for self-study a dedicated resource needs to be designed that allows the learner to control the pace and progress through the 'walk-through' process.

Solution: Using a multi-frame approach, an example program, or program fragment, is displayed in one frame where its execution can be simulated by highlighting each section of the code in execution order. In a second window, the effect of the program execution is demonstrated via an animation and in a third one, an explanation of the execution is given.

The animation illustrates the code being executed by highlighting the appropriate parts of the code that are being executed. At the same time the animation is also stepped through and the text window changes at each step to describe the action of the particular line, or section, of the program code. Thus there are three parts of the page that are cycled through in synchronisation: the code, the commentary and the animation.

The learner controls the operation via a button in the explanation window: this starts the synchronised animation.

Practice page

Context: The learner is now familiar with the concept and how it is implemented in the programming language that is being learnt. However, learners need to be able to write their own programs and this often requires a great deal of attention to detail: a simple typing error will cause an error.

Problem: New programmers often find the strict syntax requirements a block to understanding: simple syntax errors render a program useless and can cause frustration. This frustration and the time needed in the correction of syntax errors, that are not strictly to do with the learning of the new concept, reduce the effectiveness of the learning experience.

Solution: Provide a simple method of creating a syntactically correct program without the need for typing a real program. This avoids the problem of syntax errors. The learners can now be sure of their understanding of the new concept and are in a better position to create new programs of their own.

Conclusion and Future Work

The analysis of the original learning objects has resulted in the identification of a pedagogical pattern that represents a simple framework into which a series of activities can be located. The activities themselves are also patterns and are implemented as templates into which appropriate content can be inserted.

These patterns represent a simple pattern language for learning objects that support the teaching and learning of new programming concepts. They also provide reusable and customisable guidance in the design of new learning resources that is a contribution to the unmet demand referred to earlier. They may be used to produce entirely new resources or, alternatively, existing resources based upon the patterns can be customised by, for example, re-ordering activities, or by the inclusion of new activities to augment or replace existing ones. This customisation provides the opportunity for learning objects based upon these patterns to be adapted for different requirements.

Part of the work resulting from this study will be to design learning object authoring tools that utilise the patterns described above and that will allow learning designers to quickly create new resources from the patterns that have been discovered, or to customise existing learning objects for their own purposes (Boyle, 2006).

Although the patterns, here, are only concerned with the design of learning objects to support the teaching and learning of programming, it can be envisaged that similar techniques may be used to produce patterns language and tools that support other subject domains, different technologies and different pedagogical approaches. For example, Koohang and Harman (2005) strongly link online learning and constructivist pedagogy, as does Nash (2005). Learning object designs that are based upon constructivist principles could be excellent sources of design patterns that would enable learning object authors to more easily produce pedagogically sound learning resources. Additionally, Cebeci and Tekdal (2006) discuss the use of podcasts as learning objects and conclude that they need to be redesigned for instructional use; the potential for the use of learning object patterns, that incorporate the appropriate pedagogical structures, is clear.

Thus the scope of learning object patterns is potentially far broader than the case study cited here. Future work will address the technique to other subject areas and pedagogical approaches.

References

Alexander, C., Ishikawa, S. & Silverstein, M. (1977). *A pattern language: Towns, buildings, construction.* Oxford: Oxford University Press.

Boyle, T. (2003). Design principles for authoring dynamic, reusable learning objects. *Australian Journal of Educational Technology*, *19* (1), 46-58

Boyle, T., Bradley, C., Chalk, P., Jones, R. & Pickard P. (2003). Using blended learning to improve student success rates in learning to program. *Journal of Educational Media* (Special Edition on Blended Learning), *28* (2-3), 165-178

Boyle, T. (2006). The design and development of second generation learning objects. E. Pearson & P. Bohman (Eds.), *Proceedings of Ed-Media 2006 World Conference on Educational Multimedia, Hypermedia & Telecommunications* (pp. 2-11), June 26-30, 2006; Orlando, Florida.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture.* John Wiley.

Cebeci, Z. & Tekdal, M. (2006). Using podcasts as learning objects. *Interdisciplinary Journal of Knowledge and Learning Objects*, *2,* 47-57. Available at http://www.ijklo.org/Volume2/v2p047-057Cebeci.pdf

Downes, S. (2001). Learning objects: Resources for distance education worldwide. *International Review of Research in Open and Distance Learning*, *2* (1).

EASA. (2004). European Academic Software Awards (EASA) website. Retrieved January 18, 2007, from http://www.bth.se/llab/easa.nsf

Jones, R. (2004). Designing adaptable learning resources with learning object patterns. *Journal of Digital Information*, *6* (1), Article No. 305.

Kolb, D. (1985). *Experiential learning: Experience as the source of learning and development*. Prentice Hall.

Koohang, A. & Harmon, K. (2005). Open source: A metaphor for e-learning. *Informing Science Journal*, *8*, 76-86. Available at http://inform.nu/Articles/Vol8/v8p075-086Kooh.pdf

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software.* Addison-Wesley.

Goodyear, P. (2005). Educational design and networked learning: Patterns, pattern languages and design practice. *Australian Journal of Educational Technology*, *21*(1), 82-101. Available at http://www.ascilite.org.au/ajet/ajet21/goodyear.html

Nash, S. (2005) Learning objects, learning objects repositories, and learning theory: Preliminary best practices for online courses. *Interdisciplinary Journal of Knowledge and Learning Objects*, *1,* 217-228. Retrieved from http://ijklo.org/Volume1/v1p217-228Nash.pdf

Polsani, P, (2003). Use and abuse of reusable learning objects. *Journal of Digital Information*, *3* (4), Article No. 164, 19 February. Available at http://jodi.ecs.soton.ac.uk/Articles/v03/i04/Polsani/

Biographies

Ray Jones is a senior member of staff in the Department of Computing, Communications Technology and Mathematics at London Metropolitan University. He is also an active member of the Learning Technology Research Institute, which is one of the major research centres in the University. His main research interests are in the viability of applying software engineering principles to the development of learning objects.

Professor **Tom Boyle** is the Director of the Learning Technology Research Institute at London Metropolitan University. He is also the Director of the Centre of Excellence in Teaching and Learning (CETL) for Reusable Learning Objects, based on a partnership of London Metropolitan University, the University of Cambridge and the University of Nottingham.