Working with Pedagogical Patterns in PACT: Initial Applications and Observations

Andy Carle, Michael Clancy, and John Canny
Berkeley Institute of Design & EECS Department
University of California
Berkeley, CA 94720-1776
{acarle, clancy, jfc}@cs.berkeley.edu

ABSTRACT
We present several interesting applications for the Pattern-Annotated Course Tool (PACT) and pedagogical design patterns in the process of curriculum design. PACT is a visual editor in which content designers can create visual representations of their courses and annotate them with references to educational theory in the form of pedagogical patterns. Each usage scenario illustrates the opportunities for learning that PACT, the annotation process, and the artifacts that users create present to experts, novices, and everyone in between. Finally, we take an in-depth look at one ongoing course annotation being crafted by an experienced curriculum designer and examine his interesting findings from the process. An analysis of these discoveries demonstrates the utility of PACT in the curriculum design process.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms
Design.

Keywords
Pedagogical Patterns, Curriculum Development Toolkit, Curriculum Annotation

1. INTRODUCTION
The recognition of the importance of active learning and the creation of learner-centered environments (e.g. [5, 6, 9, 13]) are no longer cutting-edge developments. Technological improvements have facilitated the creation of new classroom platforms designed to put the focus back on the learners and away from the instructor. Examples of such systems include tablet PC-based collaboration tools [12], algorithm visualization kits [14], and labs that make use of web-based integrated learning environments [8]. The state of the art in instructional technology is catching up with modern recommendations from pedagogical specialists and educational researchers.

Unfortunately, the average university class has not followed this same trajectory. Most instructors teach in the same way their predecessors have taught for generations: standing at the front of a large lecture hall and speaking to their students. There are numerous notable exceptions, but it appears that the majority of university instructors do not make active learning or learner-centered instruction a focus of their courses. It is reasonable to doubt that these same instructors will be willing to move to a radically new learning environment (such as the technology-enhanced solutions previously mentioned) if they do not first understand and appreciate the basis of modern pedagogy. Worse, instructors who attempt to teach in such environments without proper training are likely to misuse the affordances presented by the technology.

There are a number of extant resources to help bring an instructor up to speed with modern pedagogy. Books and websites on the topic are plentiful, discipline-specific research groups such as SIGCSE aggregate knowledge on pedagogical practices, and a number of well designed courses are offered every day at universities around the globe. In fact, versions of these resources are often present at the home institution of an instructor. However, many teachers (particularly professors who are focused on research) lack the ability or motivation to synthesize these resources and implement successful learner-centered pedagogy in their own classroom.

In this paper, we present the Pattern-Annotated Course Tool (PACT), a visual editor designed to unify a number of curriculum design tasks under a common platform that pushes the user towards best practices in pedagogy. The focus of PACT is on concrete representations of course materials that are annotated with references to pedagogical patterns, abstractions of best practices in pedagogy and educational theory. PACT can be used by expert course designers to explore their own designs and critically analyze the theoretical support for the decisions they made in the design process. Pattern-annotated courses can be used as highly relevant and concrete learning tools by novice instructors who are taking their first steps into modern pedagogy. And, finally, PACT can be used as a visual focal point of presentations and discussions on pedagogy and curriculum design.

SIGCSE'07, March 7–11, 2007, Covington, Kentucky, USA.

2. PEDAGOGICAL DESIGN PATTERNS

The concept of design patterns has been prevalent in literature since the architect Christopher Alexander and colleagues first explained the concept in their seminal book of 1977, *A Pattern Language*: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the 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" [1]. Alexander's intent for patterns was to capture expert knowledge in a way that could be comprehended and appreciated by individuals with considerably less knowledge of the field of study. This notion has subsequently been extended to other fields, including software engineering with Gamma, Helm, Johnson, and Vlissides's famous software engineering text, *Design Patterns: Elements of Reusable Object-Oriented Software* [11].

The Pedagogical Patterns Project [16] was started in an effort to duplicate the success of these projects within the space of curriculum design. The principal idea is to identify outstanding teaching practices and record them in a pattern format that facilitates a common vocabulary for pedagogical research and practice, is accessible to novice instructors, and encourages the repurposing and reuse of techniques that are solidly grounded in modern pedagogy. The early focus of the project was on teaching object-oriented design concepts, but subsequent work by Joe Bergin [3] and others has extended the focus to other areas of computer science instruction.

In the spirit of Alexander patterns, the pedagogical patterns of the Pedagogical Patterns Project follow a consistent format:

- A description of the problem
- The forces governing the application of the pattern
- A description of the solution
- Advice on how to implement the pattern

With this information in hand, a novice instructor or a newcomer to a particular style of teaching (e.g. with tablet computers for each student) should be able to determine if the pattern applies to her situation. Ideally, the information presented in the pattern should be general enough that the instructor can tailor it to her own needs without negatively altering the pedagogical theory.

Despite this potential, pedagogical patterns have not yet seen widespread adoption. Helen Sharp and other members of the Pedagogical Patterns Project commented in 2003, "During the life of the project, we have learned a lot about patterns and their application to pedagogy, and the work is still growing and changing. … For people outside the project who don't know the material as well as members of the project, it can be quite daunting to pick up a [pattern] language and begin to use the patterns it contains" [15]. Similarly, Fincher and Utting [10] advocate a general expansion of the pedagogical patterns movement to collect additional patterns, pattern instantiations, and feedback on the effectiveness of each pattern. This expansionist attitude has guided our development of the PACT editor.

3. THE PACT EDITOR

We seek to make the world of pedagogical patterns more accessible and engaging to a wide audience. The key to our approach is the interface of the PACT editor. The software is intended to be engaging, fun to use, and valuable in the design process. Our design is shaped at the highest level by principles from the learning sciences, including making thinking visible, scaffolding understanding, and progressing from concrete representations to abstract knowledge. In this section, we describe the features of the PACT editor at a high level in just enough detail to adequately prepare for the remaining sections of this paper. For a more detailed explanation of the editor's features and the motivation behind the design, please refer to our prior report: "PACT: A Pattern-Annotated Course Tool" [7].

3.1 Course View

The primary perspective in PACT is course view (see Figure 1), in which collections of learning objects or course activities can be arranged into temporally sequenced groups representing the in-class delivery of the curriculum. The innovative feature of PACT is the ability to annotate these course activities with references to pedagogical patterns. The resulting artifact, a pattern-annotated course, is a reification of both the curriculum and the instructional expertise that was needed to create it. This direct connection between theory and real-world practice allows PACT to present the knowledge of experts in a way that is useful to other course designers.

The course view is designed to be exceptionally intuitive to navigate and manipulate. The interface is a Zoomable User Interface (ZUI) built on the Piccolo ZUI toolkit [2]. This powerful visualization makes the abstract notion of a curriculum and pedagogical pattern structures much more concrete and accessible to curriculum developers.

3.2 Editing Tools

PACT includes a standard suite of editing tools to make course creation and manipulation simple. Most changes are made by

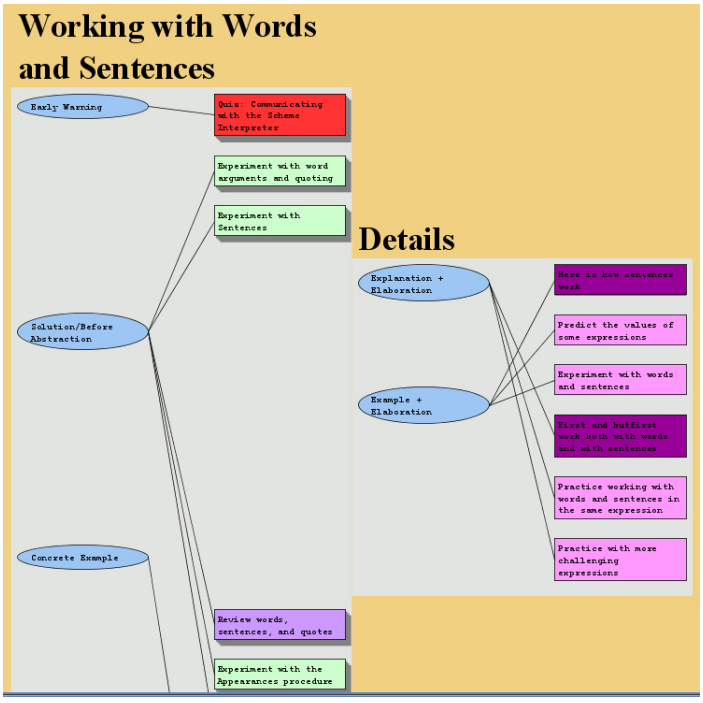

Figure 1. The second day of CS 3 annotated in course view.

direct manipulation actions performed on items in the visualization with a mouse or tablet pen. This simple interaction mechanism encourages experimentation and provides an incentive to use PACT for instructors attempting to adapt existing courses to their own needs. Basic tools such as cut/copy/paste are included for usability reasons and to ease users into the system.

Often, users will see interesting curriculum design choices in another instructor's course that they would like to use in their own design. In addition to simply copying the names and descriptions of the patterns that are being instantiated in the course of interest, it is often useful to copy the entire structure associated with that instance of a pattern. For example, the "Early Warning" pattern [4] might be associated with a sequence of assessment activities such as quizzes or homework. When moving that pattern instance into a new course, PACT will bring along the activities that it was attached to, helping preserve the semantics of the original instantiation. We refer to this process as pattern cloning.

4. APPLICATIONS

We believe that PACT is useful for both inexperienced and experienced course designers. Additionally, we see tremendous value in the use of PACT as a general curriculum creation tool that helps to stimulate serious thought about content and structure. In this section, we describe several scenarios in which the use of PACT and pedagogical patterns can shed new light on the design process and underlying educational theory.

4.1 Annotation by Expert Course Authors

Expert content developers have acquired tremendous knowledge of what works and what doesn't work when creating curricular materials. While some of this awareness may have been explicitly trained via formal study of pedagogy or educational research, often the bulk of an instructor's ability comes from the tacit knowledge built from years of honing her craft in a classroom. Our experiences have shown that it is often difficult for an instructor to explain precisely why she has made a design decision – the complexly nuanced solution was often just the first thing that came to mind at the time.

We have found that the process of annotating a course with references to pedagogical patterns helps experienced instructors unravel their own understanding of their design. As discussed at length in the next section, carefully reflecting on each portion of a course with the pedagogical patterns framework in mind can lead to an emergent picture of how the designer views the material in the curriculum along with the affordances of the learning environment in which it is being presented. The mechanisms provided in PACT help to organize and record the expert's discoveries into a meaningful, cohesive picture. Intuitive navigation and a highly visual feel to the process encourage the user to explore the annotated structure she has created and can lead to a much richer understanding of the original content creation process.

This rich understanding can, in turn, be used to iteratively improve the material and structure of the course, even if it was well designed to start with. In this usage, PACT essentially serves as a metacognition tool to help structure the instructor's design process over the course of many offerings of the same curriculum.

4.2 Content Creation by Novice Instructors

The artifacts created by expert course authors are interesting for many reasons beyond the annotation process itself. These pattern-annotated courses are rich resources for novice instructors in the process of learning a new pedagogy or curriculum platform. Each course is a real-world structure that the new instructor can relate to directly. She has likely taken a course very similar to the one annotated or may even be familiar with the expert instructor that designed and annotated it. For these reasons, the pedagogical patterns used in the curriculum take on a concrete meaning that can not be achieved in the abstract.

The real power of PACT is apparent when a novice instructor begins creating content for a new course. Initially, she is unlikely to understand the patterns that are presented alongside the course material. However, the mechanism of pattern cloning makes it easy to copy large pieces of other instructors' curricula that appear interesting. The course author can then make the content changes necessary to adapt to the new course without fundamentally changing the semantics of the pattern instances. As time goes on, we hope that the author will become increasingly curious about the patterns that she is using in her course. She can then query PACT for a much richer description of each pattern, its structure, and the underlying theory behind it. In this way, we can now teach the novice instructor about modern pedagogy using course contents that *she designed* – a highly motivating example.

4.3 Mediation for Discussion

As a highly visual medium, the PACT interface makes an excellent visual aid for describing and discussing issues in pedagogy and curriculum design. In this use, PACT is a teaching tool in the hands of the pedagogical expert. She can review her annotation with other researchers (both novice and experienced) to elicit ideas and stimulate discussion of improvements to content and structure. The experienced instructor can also use this shared artifact to help teach novice instructors or graduate students the fundamentals of modern pedagogy.

In our own research group, we have recently used PACT in this capacity to structure the iterative improvement of our basic data structures course. We have found that the eye-catching visuals of PACT help to keep the entire group focused and talking about the same issue. Additionally, the nature of the ZUI provides a great deal of context to situate the activity we are discussing in the course as a whole. There is no need to spend a lot of time discussing where a particular activity falls into the sequence or how it is related to other activities. In pattern-annotated courses, there is also very little need to discuss *why* an activity exists, as the patterns help to describe this visually.

5. SAMPLE ANNOTATION PROCESS

In this section, we will describe a model annotation process conducted by an experienced course designer and the lessons learned from the process. The course of interest is our introductory Scheme-based programming course for non-computer science majors, CS 3. CS 3 is delivered in the UC-WISE integrated learning environment [8] which provides a variety of tools for student interaction.

Table 1. A partial day's worth of CS 3 activities with their UC-WISE Step Types

Step #	*Type*	*Description of Activity*
1	Quiz	Define a function, then determine the value of an expression involving nested function calls
2	Web-text (explanation)	Introduce the terms "word" and "sentence" and the functions `first` and `butfirst`
3	Gated collaboration	Why does `(first mike)` produce an error, while `(first (quote mike))` returns "m"?
4	Web-text (explanation)	We note that numbers are self-evaluating.
5	Gated collaboration	Given functions `(define (initial1 name) (first name))` and `(define (initial2 name) (first (quote name)))` try to use each to find Mike's first initial by supplying "mike" as an argument. Explain the results.
6	Web-text (explanation + exercise)	We explain the `word` procedure, provide an example, and ask for a function `plural` that returns an "s" onto the end of its argument.
7	Web-text (explanation)	We describe the usage of the quote mark to abbreviate the `quote` function.
8	Scripted assessment	Predict the results of several uses of `first` and `butfirst`.
9	Web-text (explanation)	We explain the `sentence` procedure, and provide some examples.
10	Scripted assessment	Predict the results of several uses of `sentence` and `word`.
11	Scripted assessment	Fill in the blanks in a given collection of expressions to get specified results.
12	Web-text (explanation)	We explain the difference between using `first` and `butfirst` with sentences and using them with words.
13	Scripted assessment	Without the Scheme interpreter, evaluate several expressions using `first` and `butfirst` with words and sentences.
14	Scripted assessment	Fill in the blanks in a given collection of expressions to get specified results.
15	Gated collaboration	Explain the difference between `(first 'mezzanine)` and `(first '(mezzanine))`, and between `(first (square 7))` and `(first '(square 7))`.
16	Gated collaboration	Explain the difference between `(butfirst 'x)` and `(butfirst '(x))`.
17	Web-text (exercise)	Supply parentheses and quotes in a sequence of words to result in an expression that evaluates to a given value.
18	Gated collaboration	Experiment with a built-in function, finding out how many arguments it takes and what it returns.
19	Discussion	What are good ways to experiment with a function?

These tools include:

- pages of Web text, sometimes used for explanation or examples, sometimes used to present a programming exercise
- quizzes, for which answers and explanations are provided by the lab instructor
- scripted assessments, with which hints can be provided for incorrect answers
- collaboration tools, which can be used either as formative assessments where students compare answers with one another, or as mechanisms for sharing learning strategies, approaches to problem solving, etc
- a personal journal for information and reflection

Table 1 lists the use of these tools in a portion of the second day of lab activities for CS 3. These tools and the activities that can be created with them in mind match up well with the individual steps involved in implementing a pedagogical pattern in a curriculum.

5.1 The Annotation Process

Michael Clancy, who designed the CS 3 UC-WISE curriculum in 2002, undertook an example annotation of the CS 3 curriculum using two pedagogical patterns:

- Example + Elaboration: an example is presented and one or more subsequent activities extend, analyze, or otherwise elaborate the example
- Explanation + Elaboration: a technique or construct is explained and one or more subsequent activities use, extend, analyze, or otherwise elaborate the technique

Table 2. Labeling steps in Table 1 with pattern references

Example + Elaboration	*Explanation + Elaboration*
3+5	2+6, 2+8
9+10, 9+11	12+13, 12+14

Table 3. Common instances of the sample patterns

Students are presented with …	*then they …*
an explanation of a construct or technique	experiment to solidify their understanding
"	are quizzed to verify their understanding of the explanation
"	use the construct or apply the technique
an explanation of an alternative coding method	rewrite earlier code
example code	use the code
"	rewrite the code
"	explain how the code is similar to or differs from another program segment

By systematically stepping through the curriculum, Clancy was able to identify a number of instances of each of these patterns. Table 2 lists the identified instances of these patterns in the curriculum segment shown in Table 1. Figure 1 above shows a portion of this segment annotated in PACT. In fact, instances of these patterns turned out to comprise most of the CS 3 curriculum. This surprising result helps reveal Clancy's design process as well as his thoughts on the strengths of the UC-WISE platform. The set of tools available in UC-WISE naturally affords this type of content development – a notion that future instructors designing for the UC-WISE platform might have missed if not for the creation of this artifact.

5.2 Annotation Analysis

A careful review of the annotated course has revealed several areas for more detailed and focused analysis. For instance, there were relatively few examples of curriculum segments that did *not* fit either of the example+elaboration or explanation+elaboration patterns. (An example in Table 1 is the segment 18-19, where students are first exposed to the activity of analyzing a mystery function.) This provokes an obvious question: if these patterns are ideal for this platform and course, is there a good reason for the inclusion of activities that do not fit this mold?

Once we were cognizant of this repeating pattern, a second avenue of inquiry was apparent. Many instances of these two patterns took the form of one of the examples listed in Table 3.

With the pattern in mind, it is critically important that each of the "explanation" activities be accessible to students. Otherwise, the whole sequence falls apart and the student will be lost. Similarly, the constructs, techniques, and code segments that are presented and analyzed must be sufficiently general to support the base of the sequence and be useful in later curriculum segments. If an activity fails any of these criteria, does it still merit inclusion in the curriculum or should it be replaced?

These questions have provoked an ongoing evaluation by our curriculum development group of certain activities in CS 3. This discussion would have never taken place if not for the annotation of CS 3 and subsequent analysis. We feel that similar analyses are critical to the development of robust learner-centered curricula for other courses, classroom platforms, and pedagogies.

6. CONCLUSION

We have discussed the general consensus that university classroom practices are falling behind modern learner-centered pedagogies and the instructional technologies that are designed to support them. While resources are available to train instructors on modern pedagogical practices, few take the time to synthesize and implement these ideas in their own curriculum design. We presented the Pattern-Annotated Course Tool, a learner-centered visual editor designed to bring together experts and novices in novel ways via the creation and consumption of course representations annotated with references to pedagogical patterns.

We believe that the success of the pedagogical pattern movement hinges on many instructors, expert and novice, getting involved in the creation of patterns and pattern instantiations. Towards this end, we have identified several ways in which users of various levels of expertise could use PACT and pedagogical patterns in their curriculum design process. Finally, we examined one portion of a large annotation project to show the potential value of the methods and tools involved. We believe that PACT and pedagogical patterns show great potential as the centerpiece of a reinvigorated, expanded, and distributed effort to move active learning strategies into the average university classroom.

The PACT webpage is located at:
`http://www.cs.berkeley.edu/~acarle/PACT`

7. ACKNOWLEDGMENTS

This work was supported by a Microsoft Research Tablet PC and Computing Curriculum RFP award.

Special thanks to the UC-WISE group for their feedback and encouragement throughout the PACT design process.

8. REFERENCES

[1] C. Alexander, S. Ishikawa and M. Silverstein, *A Pattern Language,* Oxford University Press, 1977.

[2] B.B. Bederson, J. Grosjean and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Transactions on Software Engineering,* vol. 30, pp. 535-546, 2004.

[3] J. Bergin, "Fourteen Pedagogical Patterns," 2002. Available: http://csis.pace.edu/~bergin/PedPat1.3.html

[4] J. Bergin, J. Eckstein, M.L. Manns and H. Sharp, "Feedback Patterns," 2005. Available: http://www.pedagogicalpatterns.org/current/feedback.pdf

[5] J. Bransford, A.L. Brown and R.R. Cocking, "How people learn: Brain, Mind, Experience, and School." 1999.

[6] A.L. Brown and J.C. Campione, "Guided discovery in a community of learners," in *Classroom lessons: Integrating cognitive theory and classroom practice,* K. McGilly Ed. Cambridge: MIT/Bradford, 1994.

[7] A. Carle, J. Canny and M. Clancy, "PACT: A Pattern-Annotated Course Tool," in Proceedings of World Conference on Educational Multimedia, Hypermedia, and Telecommunications, 2006, pp. 2054-2060.

[8] M. Clancy, N. Titterton, C. Ryan, J. Slotta and M. Linn, "New roles for students, instructors, and computers in a lab-based introductory programming course," in SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education, 2003, pp. 132-136.

[9] P. Dillenbourg, *Collaborative Learning: Cognitive and Computational Approaches,* Oxford, UK: Elsevier Science Ltd., 1999.

[10] S. Fincher and I. Utting, "Pedagogical patterns: their place in the genre," in ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education, 2002, pp. 199-202.

[11] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns -- Elements of Reusable Object-Oriented Software,* Reading, Massachusetts: Addison-Wesley, 1995.

[12] M. Kam, J. Wang, A. Iles, E. Tse, J. Chiu, D. Glaser, O. Tarshish and J. Canny, "Livenotes: a system for cooperative and augmented note-taking in lectures," in CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems, 2005, pp. 531-540.

[13] E. Mazur, *Peer Instruction: A User's Manual,* Upper Saddle River, NJ: Prentice-Hall, 1997.

[14] T.L. Naps, J.R. Eagan and L.L. Norton, "JHAVÉ—an environment to actively engage students in Web-based algorithm visualizations," in SIGCSE '00: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, 2000, pp. 109-113.

[15] H. Sharp, M.L. Manns and J. Eckstein, "Evolving Pedagogical Patterns: The Work of the Pedagogical Patterns Project," *Computer Science Education,* vol. 13, pp. 215-330, 2003.

[16] H. Sharp, M.L. Manns and J. Eckstein, "The pedagogical patterns project (poster session)," in OOPSLA '00: Addendum to the 2000 proceedings of the conference on Object-oriented programming, systems, languages, and applications (Addendum), 2000, pp. 139-140.