

PACT: A Pattern-Annotated Course Toolkit

Andy Carle

**Prepared for Human-Centered Computing with Professor John Canny
Spring Semester, 2005**

1. Introduction

From the first primitive individual that passed along knowledge of how to use a tool to the medieval training of priests and monks to the classroom of a modern research institution, education has played a vital role in shaping human progress and society. Over this time span the contents of the educational messages and the mechanisms used to deliver and structure them have steadily evolved. As early as the 1770s, pedagogy was being viewed as a science meriting extensive study [1]. The works of Lev Vygotsky (early 1900s) and Jean Piaget (mid 1900s), among others, added a level of formalism to the study of human learning. The societal acceptance of lifelong learning (continuing education among adults) over the past fifty years has further solidified the role of education in the modern world. A strong and well-rounded education is now seen as a requirement to compete and survive in the modern world with significance that rivals even that first early human who taught the next generation how to use tools to hunt.

As the societal call for education has increased, so too has interest in educational research. Pedagogical studies have vastly grown in number and scope in recent decades. As a result, a great deal of insight into the proper structure, contents, and delivery of knowledge has been gained. Unfortunately, most educators have been slow to pick up on relatively new techniques: today's classrooms look very much like those that Vygotsky himself learned in nearly a century ago. This academic stagnancy is troubling, particularly to those who have seen just how much better a course can be when designed and carried out with modern pedagogical research in mind. Fortunately, recent research in Human-Computer Interaction combined with educational research suggests that technology can help bridge the gap between what educators are currently doing and what researchers feel they should be doing. The rest of this paper will put into context and describe my contribution to one such piece of technology, the PACT (Pattern-Annotated Course Toolkit).

2. Context

2.1 Learner-Centered Education

“At the heart is the idea that people learn best when engrossed in the topic, motivated to seek out new knowledge and skills because they need them in order to solve the problem at hand” [2]. This statement describes the driving force behind a gradual revelation in pedagogical research called learner-centered education. Traditional curricula are focused around the content of the course. Broad subjects are divided apart from each other and broken down into “manageable” groups that are passed along to the student via the execution of a lesson plan. Learner-centered courses are instead focused around a (often small) set of realistic problems designed to motivate the student to want to learn about a subject on their own. With this change of organization, the student transforms from a passive learner that sits in lectures and reads textbooks to an active learner that solves problems through their own insight, research, and reflection. The use of shared artifacts, from chalkboards and flashcards to modern software systems like UC-WISE [3], is often helpful in this development of a student into an independent learner.

These shared artifacts, along with the problems that motivate learning, are called learning objects. Well-structured courses built around learning objects have shown remarkable improvements in student learning. But this structure does not come automatically with the presence of learning objects. It must be carefully crafted by an expert educator to be fully effective. [4]

2.2 Academic Stagnancy

Despite the plethora of evidence supporting learner-centered instruction, university professors are largely uncomfortable with designing curricula that deviate significantly from traditional lectures. The reasons for this aversion to new course formats are varied. First, many professors are unfamiliar with how such a course should be run. Not only have they never taught a course like this, most have never taken or seen a course built around learner-centered environments. Second, many professors are uncomfortable with the idea of decentralizing themselves in the course content. This concern is amplified by a lack of understanding of just what their role would be in such a course. Finally, it is likely that professors are concerned about the work involved to start from scratch re-designing a course. The culture in much of academia (including Computer Science) emphasizes research while pushing faculty away from pedagogical concerns. Therefore, if designing a course around learner-centered methodology is more time consuming than sticking to the traditional course design methods it is unlikely that anyone will bother doing so. [4]

2.3 Pedagogical Patterns

Pedagogical researchers have noticed that there are a variety of clear and repeatable patterns in education, both traditional and learner-centered. To make this structure more

accessible to the general population of educators, these patterns can be identified, formalized, and shared.

Pedagogical patterns fit the general structure defined by Christopher Alexander almost thirty years ago: “Each pattern describes a problem that 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” [5]. That is to say, a pattern pre-defines potential solutions to recurrent design problems, regardless of the context of a specific instance of the problem. Identifying and elaborating on such a pattern depends on the existence of a pattern language that allows the clear expression of the “big picture” idea behind the pattern. These abstract pattern definitions should be able to serve as a jumping off point for insight into the particular instance of a problem, leading way to an original solution fit to the context of the instance.

The Pedagogical Patterns Project [6] is one group that is working on pattern languages for pedagogy and to identify major pedagogical patterns. (See Appendix A – A Sample Pedagogical Pattern from [7] for a sample pattern, “Mistake.”) One interesting point to note from these sorts of pedagogical patterns is that they can *vary wildly in scope*. Some patterns address issues of course-wide design and flow. Others, such as “Mistake,” give a suggestion for a specific type of activity (or assignment) that could be incorporated into a class. A second key point is that these patterns are *goal oriented*. **A pedagogical pattern relates a class of learning objects or general course structures to a generalized goal.** Or, in the words of [8], “A pattern is in fact a solution to an instructional problem that is (at least partially) defined by learning goals.” It is important to keep these two points, variance in scope and goal orientation, in mind when studying pedagogical patterns.

3. Problem

The PACT (Pattern-Annotated Course Toolkit) is a potential set of tools to assist in the development and maintenance of a repository of pedagogical patterns and courses. The pattern repository supports course creation in several ways: (i) starting with an existing course, an instructor can modify elements of it such that the pattern structure acts as a guide; (ii) an instructor can start with one or several abstract patterns and combine them with learning objects to build a new course that follows good learning principles (e.g. inquiry-based, other constructivist, cooperative etc.); (iii) when an instructor has completed a course, he or she can contribute the course with its instantiated patterns to the repository; (iv) instructors can construct new patterns that they have found successful from their courses and contribute them to the repository; (v) instructors can review and rate patterns, learning objects and courses that they have made use of from the repository, and those reviews can be shared with other instructors [4].

This work required to support these actions can be roughly split into three parts: the design tool that is used to create and manipulate patterns and courses, the data schema used to represent and store patterns and courses, and the development of the repository

backend that supports review, rating, and collaboration. **For this project I have chosen to focus on the schema used to represent course data (used in items i and ii above) and the potential for PACT to serve as a learning tool (in line with item ii above).** The work of creating the visual design tool itself and building the repository have been left for other projects.

4. Approach

4.1 Data Schema

4.1.1 Course Designs as Boundary Objects

The smooth sailing of a course at the university level is dependent upon the collaboration of a wide range of professionals. This fact is abundantly clear when looking at a typical learner-centered course. The successful learning of an individual student is often dependent upon the coordinated actions of an instructor, the course teaching assistants and lab assistants, any number of tutors, the instructional computing support staff, educational software developers, and education researchers. Communication across such a potentially diverse group of disciplines can be challenging. All of the actors within this network must be kept fully “in the loop” for the course to go well. For example, if a professor is working with an education researcher to develop and implement pedagogical patterns within her course’s labs, the teaching assistants and lab assistants must be well-informed to ensure that the professor’s vision becomes a reality “on the ground” [9]. A similar situation could occur when the professor is explaining his class’s technology needs to his department’s instructional computing group. If the technical people don’t understand the “big picture” they may be unwilling to meet the seemingly extravagant computing needs of a learner-centered course. To further complicate the matter, the goal is to enable professors with vastly differing backgrounds to look at each others courses and patterns and understand them.

From the literature on communities of practice, particularly the work of Etienne Wenger [10], we gain the concept of a boundary. This sort of boundary exists between communities with different histories, cultures, and constructions of meaning. Misunderstandings are likely to arise when working and communicating across these boundaries. Wagner (in reviewing the work of Leigh Star) discusses the concept of boundary objects that “coordinate the perspectives of various constituencies for some purpose.” As this is precisely what we want to accomplish with our annotated course descriptions, it would be wise to keep the qualities of an effective boundary object in mind while developing PACT. Star identifies these qualities as: modularity: each perspective can attend to one specific portion of the boundary object; abstraction: all perspectives are served at once by deletion of features that are specific to each perspective, accommodation: the boundary object lends itself to various activities; standardization: the information contained in a boundary object is in a pre-specified form so that each constituency know how to deal with it locally [10].

Designing the PACT schema with this in mind adds some overhead to the essential goal of annotating course objects with pedagogical patterns. For example, it forces a more robust representation of learning objects, course goals, and the relationship between them. However, I believe that this robustness is essential even if you aren't interested in communication among the actors involved with one course, at least for *some* patterns. Having at least the option of thoroughly specifying the details of a learning object will help others learn just how a pattern is implemented, even if that particular implementation does not apply to the course they are designing. This is an extension of the general learner-centered principle of "concrete to abstract."

4.1.2 Educational Environment Markup Language (E²ML)

Educational Environment Markup Language (E²ML) [11] is a modeling language developed to support instructional design. Instructional design, as it is typically defined [12], focuses on learning needs, goals related to knowledge, skills, and attitude, and methods for getting students from a variety of starting points to those goals. E²ML approaches this problem with the following components:

1. **Goal Statements:** A list of all the goals of a course. These goals include both learning goals and meta-goals (e.g. staff or learning object evaluation). An individual goal definition includes the following parts: statement, a freeform text description of the goal; target, the individuals who should be moving towards meeting this goal; stakeholder, the people in charge of making sure the targets meet the goal; approach, a description of how this goal will be reached; and importance, a measure of the relative significance of this goal to the course as a whole. (See Figure 1)
2. **Overview Diagrams:** A description of the potential flow within the course. For example, E²ML supports a dependency diagram that can be used to describe course units that have logical dependencies between them (e.g. A should follow B). A highly complicated overview diagram would be a fully "zoomed-out" view of the flow of the entire course.
3. **Action Diagrams:** An action diagram in E²ML describes one course activity. The scale of these actions can vary (i.e. a lab and an individual lab exercise are both reasonable actions). In relation to PACT and learner-centered education, an action is roughly equivalent to a learning object. At a coarse level, actions are described as a set of: identifying information, data about what the action is and who is involved; initial state, the situation of the learners before this action along with the inputs to the action; final state, the desired outcome of the action and the outputs; and action performance, a description of how the activity is to be performed. (See Figure 2)

| GOAL STATEMENT | | | | | |
|----------------|--|--------|-----------------------|------------------------------------|------------|
| TAG | STATEMENT | TARGET | STAKEHOLDER | APPROACH | IMPORTANCE |
| A1 | Recognize critical success factors in communication | All | Head | Case studies and discussion | 5 |
| A2 | Analyze successful and unsuccessful communications | All | Head | | 4 |
| A3 | Recognize differences between direct and mediated communication settings | All | Head | | 5 |
| B1 | Recall key concepts of communication | All | Head | Critical discussion on movie clips | 4 |
| C1 | Perform effective videoconferencing | All | Head | Guidelines, examples and exercises | 2 |
| C2 | Perform effective audioconferencing | All | Head | | 3 |
| C3 | Write effective emails | All | Head + Mktg. Director | | 5 |
| C4 | Deliver effective presentations | All | Head + Mktg. Director | | 5 |
| C5 | Effectively integrate corporate Web sites into communication | All | Mktg. Director | | 3 |

Figure 1 – A Goal Statement in E²ML

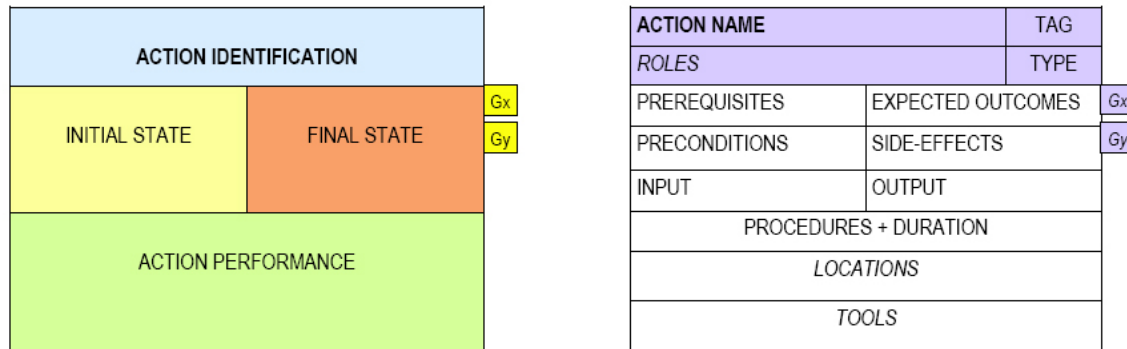


Figure 2 – An Action Diagram in E²ML

E²ML offers the advantages of being well-studied and having been designed with communication among stakeholder as an explicit objective. E²ML course descriptions have been shown to be easy to understand, even if you are not intimately familiar with any specifics of a particular course. However, the level of detail mandated by E²ML is a bit much for someone new to course development. In addition, E²ML makes no explicit references to pedagogical patterns at this time. The relationship between course goals and the overview diagrams which, if present, would be a sort of pattern description are in reality not clear. E²ML is a good starting point for a PACT schema, but it does not cover all the issues.

4.1.3 XML Schema

To best capture the structure of a course, facilitate the building of relationships between patterns and learning objects, and allow easy extensibility it appears that a reasonable course to follow is to very loosely implement E²ML and pattern-related extensions in XML [13]. XML has many virtues, such as being easy to manipulate, extend, read, and parse. Determining the structure of the XML schema itself requires working closely with

educational researchers and educators to determine exactly what needs to be represented. Through this sort of study, the schema can be iteratively designed.

4.2 PACT as a Learning Tool

4.2.1 Human Learning Principles

The works of Lev Vygotsky and those working in his tradition have given us a lot of knowledge on how humans (both children and adults) learn new skills. From this work we find that there are natural progressions through the learning process that can be encouraged by educators and, subsequently, educational software. Of particular interest here is Vygotsky's notion of the Zone of Proximal Development (ZPD). The ZPD theory states that every individual has a set of things that they can do on their own, a set of things they can do with assistance, and a set of things that they can not do even with assistance. The goal of "scaffolding" is to target tasks that the learner can do with assistance and gradually bring those tasks into the area that the learner can do on their own. Subsequently, tasks that were completely out of reach before will now be possible with assistance, and the cycle can repeat.

4.2.2 Scaffolding in PACT

Some of the ways that PACT should act as a scaffold for educators trying to learn how to develop learner-centered courses are clear. For example, viewing another educator's course, reviewing comments on it, and manipulating it to reflect your own course should be a good way to learn the general principles. What I am more interested in here is point ii of the PACT plan: "an instructor can start with one or several abstract patterns and combine them with learning objects to build a new course that follows good learning principles (e.g. inquiry-based, other constructivist, cooperative etc.)." I believe that there is a clear opening here for the PACT system to actively make suggestions and help the user understand what the strengths and weaknesses of their course design are. To this end I have implemented a simple course analyzer that provides example suggestions to the user for how to modify their course. In particular, I'm interested in what insight can be gained by comparing two courses to each other and making suggestions on how to improve one based on the other. An analysis of user reactions to these prompts can help determine whether the concept is useful or not.

5 Related Work

In addition to the work on design patterns, pedagogical patterns, and E²ML mentioned already, there are several areas of related work. I will very briefly touch on these in this section.

5.1 e-Learning Systems

The idea of using technology to deliver course materials is not a new one. The ideas of breaking a course into learning objects and using technology as a mediator for

collaboration, inquiry, and assessment have been well-studied. One such system with a direct influence on the PACT system is the UC-WISE system [3]. WISE has proven that a learner-centered course can be carried out with a large class size at a major university. It is from a study of the courses carried out in WISE that the clear need for organization, abstraction, and annotation have arisen.

5.2 Extensions to E²ML

The developers of E²ML are working on an extension to their language to support pedagogical patterns [8]. At this point it is not entirely clear what direction they will go with their potential system, but it appears that they intend to rely very heavily on the existing E²ML format with an additional system of simple annotations.

6. Achievements

6.1 Reading

I began this project by doing a review of the available material on learner-centered education and pedagogical patterns. This led me to learn about E²ML as well. This was in line with my initial plan.

6.2 Data Schema

Over the course of the semester I created and evolved the XML schema used to represent course data. I used an iterative process, during which I worked with Mike Clancy (UCB CS Instructor and educational researcher) to annotate small pieces of learner-centered courses that he teaches. In addition to working with Dr. Clancy, I attempted to break a few of the labs used in CS61c (which I teach) into learning objects and to annotate them with patterns. At each step, schema elements were added, removed, and rearranged. The current iteration of this schema (See Figures 3, 4 and 5) is a very simple design that allows for the use of E²ML components where appropriate. Patterns are represented here as simple identifiers, with the anticipation being that they will eventually actually be cross-references with a central repository as opposed to having an entire pattern language embedded within one course. Many of the structural features of E²ML are not represented here because we feel they would be better expressed in terms of patterns than using conventional E²ML mechanisms. This schema has proven effective for representing simple course descriptions (See Appendix B – A Simple XML Course File), but more work needs to be done as more complicated courses are described. This part of the project was as I had planned.

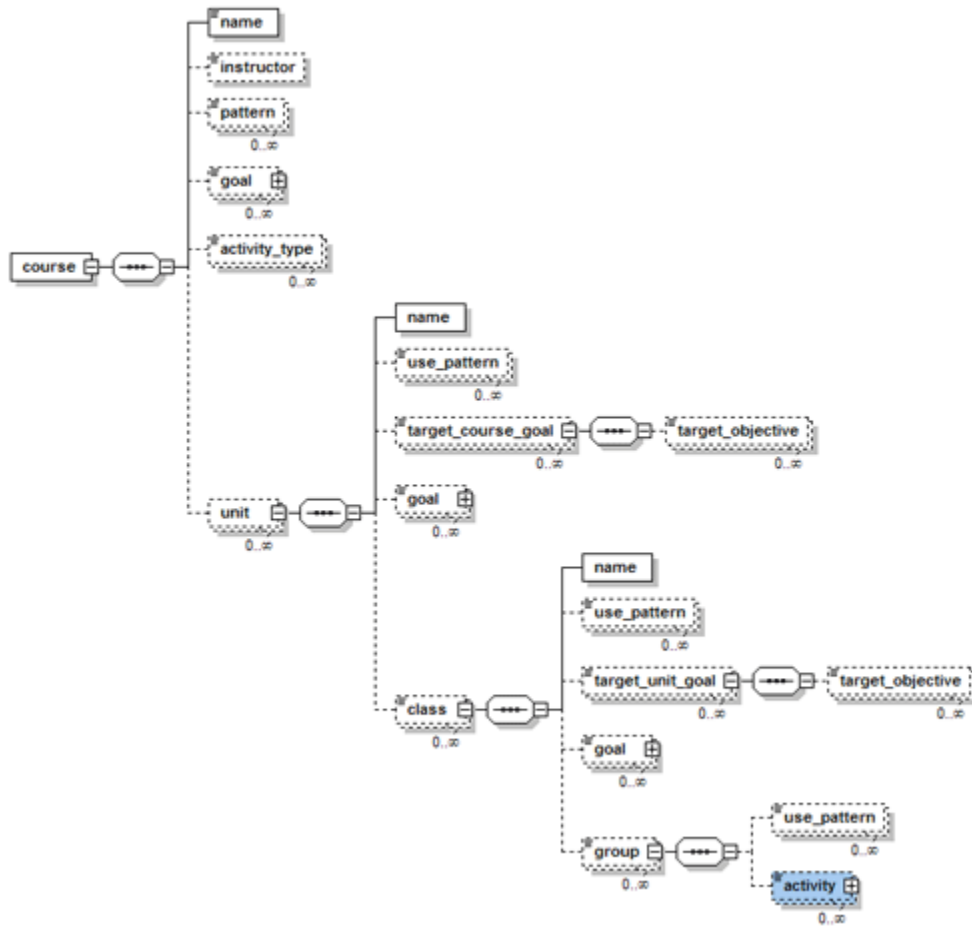


Figure 3 – The Current Iteration of the PACT XML Schema

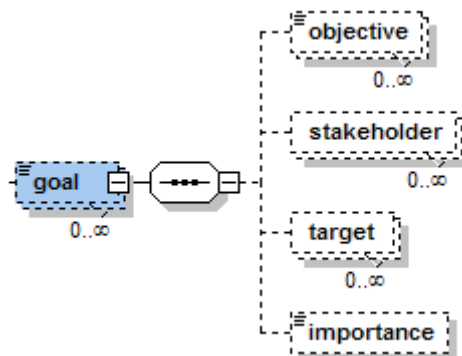


Figure 4 – Breakout of a Goal

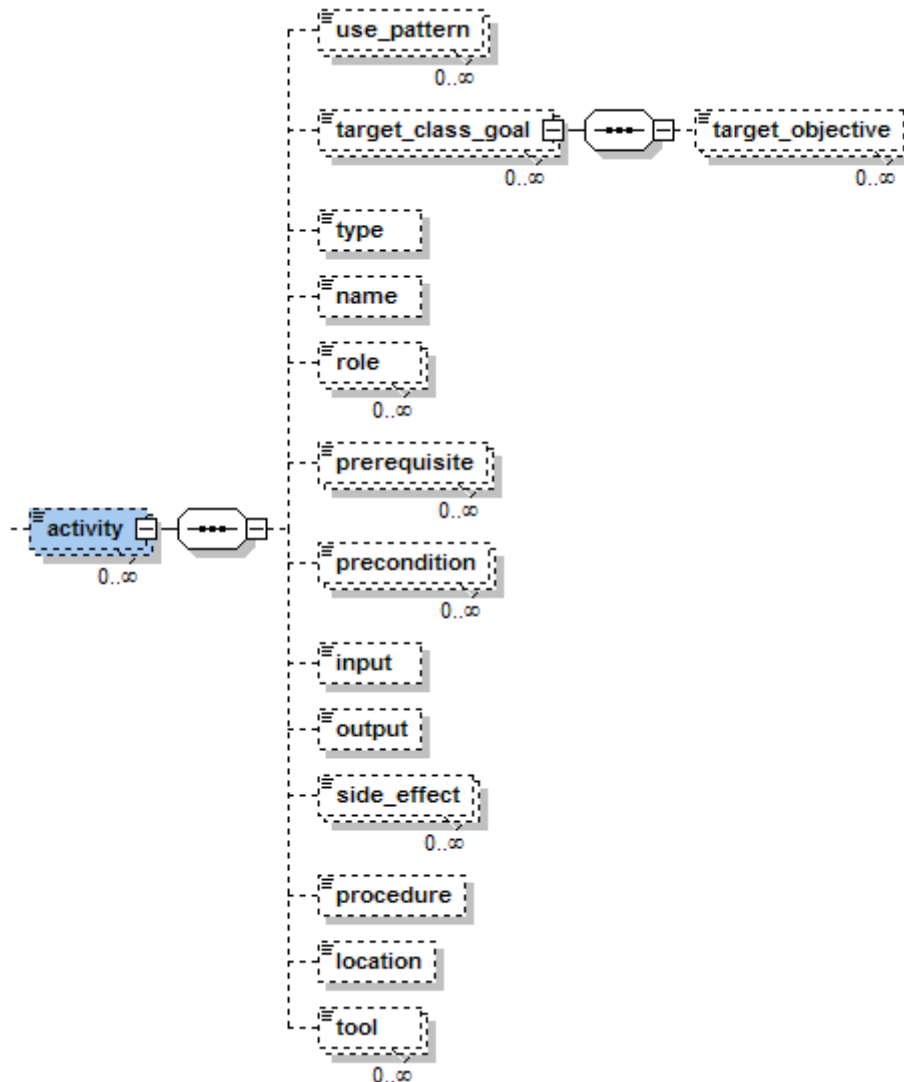


Figure 5 – Breakout of an Activity

6.3 PACT as a Learning Tool

My initial plan for the third step of this project was to begin designing the visual design tool. However, when a fellow classmate and I each reached the end of our literature reviews and decided how to break up the work he opted to work with the visualization while I focused on the data schema. I came up with the idea of working with the scaffolding aspects of PACT as a way to extend my portion of the course project as well as draw in another topic of Human-Centered Computing. So, this was the last thing that I got a chance to work on this semester.

In the little time that I had, I created a simple XML parser in Java and built in a series of basic rules with which to analyze course descriptions. What little I did build in works well and gives feedback that strikes me as potentially useful for an educator who is trying to develop her own learner-centered course. Unfortunately, my goal of evaluating the potential effectiveness of this sort of feedback was not accomplished in time for the

writing of this report. The main problems here were that 1) none of this work has yet been integrated into any sort of visual tool and 2) time constraints did not allow me to sit down and hand build an XML course representation with an educator who is inexperienced in learner-centered techniques. Despite the time crunch, I believe that this work could eventually expand to become something very useful.

7. Findings

7.1 Reading

I found the literature on pedagogical patterns to be quite interesting. I'm very interested in teaching, and much of this material really struck home for me. Based on my readings here I've begun remodeling some small parts of CS61c, which I will be instructing Summer Semester 2005. I also found the dissertation on E²ML to be very enlightening from a pedagogical standpoint, even if it wasn't always directly related to the PACT tool.

7.2 Schema Design

My over-arching finding here was that simpler (or at least the option of simplicity) is better for this application. Working with a very complicated schema proved frustrating, particularly when working by hand. Forcing every component of E²ML on the user would clearly be too much. But, despite that, I still firmly believe that there are some patterns that cannot be clearly expressed without a relationship to the details of a course activity. While the general concept of the pattern may be lucid enough without a concrete example, if a knowledgeable user is attempting to discover all of the patterns that they use in instructional design I believe that they need the concrete elements of their own course to work with. An example of this type of pattern is the Mistake pattern (See Appendix A) which I doubt could have ever been discovered without a review of the individual activities that make up that particular educator's courses. This also shows up very frequently with patterns that are dependent upon a particular structure of course goals. If the goals of the course are not made visible I find it unlikely that such patterns would have ever emerged.

7.3 PACT as a Learning Tool

I had the least time to experiment and to reflect on this part of the system. However, at a cursory glance the output of such a tool would appear useful. The ability to compare your own course with a known high-quality course provides some compelling insight. Unfortunately, at this point this is just my own instinct as a teacher. Much more experimentation (on a much more robust system) is needed to truly determine the value of this type of feedback.

8. Conclusion & Future Work

The PACT tool has the potential to help usher in a dramatic change in education systems throughout the world. As more and more educators encourage active learning within learner-centered environments the quality of pedagogy and student learning should increase dramatically. There are many challenges to getting a tool like PACT off the ground, but through work done this semester (both in my own project and others') a foundation for future work has been built. Continued evaluation and modification of the data schema will be required as the project goes forward. Integration of the data representation and a visualization will be essential as work continues. If work is to continue on the automatic scaffolding system it will have to incorporate much more sophisticated mechanisms from artificial intelligence and pattern recognition. But, despite these complications I believe that there is a bright future for PACT, built upon work completed this semester.

References

- [1] *Education*. Wikipedia: The Free Encyclopedia, 2 May 2005, 21:46 UTC.
<<http://en.wikipedia.org/wiki/Education>>
- [2] Norman, D.A. & J.C. Spohrer (1996): *Learner-centered Education*. Communications of the ACM, Vol. 39, No. 4 (24-27)
- [3] Linn, M.C. & M.J. Clancy: *Overview of the UC-WISE Project*.
<<http://www.cs.berkeley.edu/~clancy/UCWISE.overview.html>>
- [4] Canny, J.F. et al. (2004): *PACT Proposal*.
- [5] Alexander, C, Ishikawa, S. & Silverstein, M. (1977): *A Pattern Language*. Oxford, UK: Oxford University Press
- [6] Sharp, H. et al. (2000): *The Pedagogical Patterns Project*. OOPSLA 2000 Addendum.
<<http://www.pedagogicalpatterns.org/>>
- [7] Bergin, Joseph (2000): *Fourteen Pedagogical Patterns*. Proceedings of EuroPLoP 2000.
- [8] Belfer, K. & Botturi, L. (2004): *Online Learning Design with Pedagogical Patterns*. SALT Orlando Conference 2004
- [9] Eckstein, J., et al. (2002): *Patterns for Active Learning*.
<<http://www.pedagogicalpatterns.org/current/activelearning.pdf>>
- [10] Wenger, E., et al. (1998): *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press (July 28, 1998)
- [11] Botturi, L. (2003): *E²ML Educational Environment Modeling Language*. 2003 PhD Dissertaion: Universita della Svizzera Italiana, Lugano, Switzerland
<<http://www.istituti.usilu.net/botturil/web/e2ml/>>
- [12] *Instructional Design*. Wikipedia: The Free Encyclopedia, 3 May 2005, 21:56 UTC.
<http://en.wikipedia.org/wiki/Instructional_design>
- [13] *Extensible Markup Language*, World Wide Web Consortium. <<http://www.w3.org/XML/>>

Appendix A – A Sample Pedagogical Pattern from [7]

Mistake

(Version 2.1, July 2000)

Students are asked to create an artifact such as a program or design that contains a specific error. Use of this pattern explicitly teaches students how to recognize and fix errors. We ask the student to explicitly make certain errors and then examine the consequences.

PROBLEM/ ISSUE

People make mistakes. Students often don't know how to interpret the error messages provided by their tools or what to do to solve problems that are diagnosed by the tools. Debugging is an essential skill, whether done with a sophisticated debugger, or just by comparing actual outputs with expectations. Students don't initially know what occurs when errors are made and so don't know what to do when they see program diagnostics and incorrect outputs.

AUDIENCE/ CONTEXT

This is very applicable to the early stages of learning programming. Syntax and semantic errors are frequent and students need to become familiar with the messages produced by compilers and run-time systems and what they indicate about the program.

The pattern could also be used in an analysis or design course in which certain specific, but fairly common, errors are required to be part of the design. The instructor can provide a part of the design, which is flawed in some way, and the students are asked to complete the design without changing the instructor's part. (See the [Fixer Upper](#) pattern also.)

This pattern is often used in Database and Operating Systems courses where the students are asked to explore the conditions leading to deadlock, by specifically causing it. Similarly, in learning concurrent programming, students are often asked to write programs in which race conditions between processes lead to the corruption of data.

FORCES

Students, like professionals, make errors. Professionals generally know what the indications of an error are, but students do not.

We usually help students avoid errors, but they occur anyway. Students should have a way to explore the effects of errors so that they can recognize them by their effects.

SOLUTION

Ask students to produce an artifact with certain specific errors (usually a single error). The effect of the error is then explored.

For example, give students an assignment in which they are instructed to create and run a program with certain specific errors. Ask them to comment on the diagnostics produced and/or why no diagnostics were produced for the error.

DISCUSSION/ CONSEQUENCES/ IMPLEMENTATION

Students become more familiar with errors and how to correct them. Having seen specific errors, they can learn to avoid making them in the first place.

You should carefully prepare exercises using this pattern and be sure to do the exercise before the class does, so that unanticipated occurrences can be prepared for and you can later provide explanations of precisely what happened.

Follow up. It is important that the students get a chance to discuss interesting things that occur while making these errors.

One especially useful technique is to ask students to make errors that, when they occur accidentally, are especially hard to diagnose when made accidentally. One example of this is errors made in C++ template instantiations.

SPECIAL RESOURCES

The instructor simply needs knowledge of common errors.

RELATED PATTERNS

In [*Fixer Upper*](#) the instructor makes the errors and the students correct them.

In [*Test Tube*](#) we ask for explorations. Here we ask for explorations of specific errors.

EXAMPLE INSTANCES

In addition to those mentioned above, this pattern can be used effectively to teach students about pointers in languages like C or C++, by having them make all of the common pointer errors purposely. This particular use is somewhat dangerous on computers that have memory mapped i/o and unprotected operating systems. Both syntax and semantic errors can easily be explored using this pattern.

One exercise from an old book [Teague] was to write a program that produced every diagnostic mentioned in the manuals for a given (Fortran) compiler. This is, not surprisingly, very difficult to do. Impossible, for some compilers, as the documentation and the compiler are not completely parallel.

CONAINDICATIONS

This pattern can be over used. It is best used early and in response to student questions.

REFERENCES

Computing Problems for Fortran Solution, Robert Teague, Canfield Press, 1972.

Appendix B – A Simple XML Course File

```
<?xml version="1.0" encoding="UTF-8"?>
<course>
  <name>CS 3</name>
  <instructor>Mike Clancy</instructor>
  <pattern id="1">
    Early Warning
  </pattern>
  <pattern id="2">
    Solution|Before Abstraction
  </pattern>
  <pattern id="3">
    Concrete Example
  </pattern>
  <pattern id="4">
    Explain it Yourself
  </pattern>
  <pattern id="5">
    Try it Yourself
  </pattern>
  <activity_type id="1">
    Text
  </activity_type>
  <activity_type id="2">
    Gated Collaboration
  </activity_type>
  <activity_type id="3">
    Scripted assessment
  </activity_type>
  <goal id="0">Scheme Interpreter
    <objective id="1"> Students should understand the use of delimiters in
    Scheme.</objective>
    <stakeholder>Instructor</stakeholder>
    <stakeholder>Lab TAs</stakeholder>
    <target>All Students</target>
  </goal>
  <unit num="1">
    <name>Scheme Syntax</name>
    <target_course_goal>0
      <target_objective>1</target_objective>
    </target_course_goal>
    <goal id="0">Scheme Interpreter
      <objective> Students should understand the use of delimiters in
      Scheme.</objective>
      <stakeholder>Instructor</stakeholder>
      <stakeholder>Lab TAs</stakeholder>
      <target>All Students</target>
    </goal>
    <class num="2">
      <name>Lab 2</name>
      <use_pattern>1</use_pattern>
      <use_pattern>2</use_pattern>
      <use_pattern>3</use_pattern>
```

```

<use_pattern>4</use_pattern>
<use_pattern>5</use_pattern>
<target_unit_goal>0</target_unit_goal>
<group>
  Quiz on previous day's material.
  <use_pattern>1</use_pattern>
</group>
<group>
  Experiment with word arguments and quoting
  <use_pattern>2</use_pattern>
</group>
<group>
  Experiment with Sentences
  <use_pattern>2</use_pattern>
</group>
<group>
  Review words, sentences, and quotes
  <use_pattern>2</use_pattern>
  <activity>
    Analogy to a Pez dispenser
    <use_pattern>3</use_pattern>
    <type>1</type>
  </activity>
  <activity>
    Distinguish between (first '(square x))&(first
    (square x))
    <use_pattern>4</use_pattern>
    <type>2</type>
  </activity>
  <activity>
    Distinguish between (butfirst 'x) and (butfirst '(x))
    <use_pattern>4</use_pattern>
    <type>2</type>
  </activity>
  <activity>
    Supply parentheses and quotes to get a given result
    <use_pattern>5</use_pattern>
    <type>3</type>
  </activity>
</group>
<group>
  Experiment with the "appearances" procedure
  <use_pattern>2</use_pattern>
</group>
<group>
  Think about evaluating expressions with quotes
  <use_pattern>2</use_pattern>
</group>
<group>
  Use a sentence to "package" information
</group>
<group>
  Consider some typical student misconceptions
</group>
<group>
  Review new vocabulary and scheme procedures

```

```
</group>
<group>
  Homework
</group>
<group>
  Follow-up lecture activity: play "Who's on first?"
  <use_pattern>3</use_pattern>
</group>
</class>
</unit>
</course>
```