Promoting Learning of Instructional Design via Overlay Design Tools

Andy Carle

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-246
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-246.html

December 14, 2012

Promoting Learning of Instructional Design via Overlay Design Tools

By

Andrew Jacob Carle

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division
of the
University of California, Berkeley

Committee in charge:

Professor John Canny, Chair
Professor Marcia Linn
Senior Lecturer Michael Clancy

Fall 2012

Promoting Learning of Instructional Design via Overlay Design Tools

Abstract

Promoting Learning of Instructional Design via Overlay Design Tools

by

Andrew Jacob Carle

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Canny, Chair

Design is a notoriously difficult profession to practice, and it is even more difficult to learn. Traditionally, learning of design skills has been situated in the context of apprenticeships or formal design studios. Unfortunately, these methods are inaccessible to practicing professionals due to constraints on time and location. And, indeed, professional designers must continuously update their knowledge as paradigm shifts in design practice threaten to make their skills obsolete. An ideal resolution to this problem is to situate the learning of design skills within the professional practice of design. This dissertation studies an approach to this mode of situated learning, focusing on integrating learning mechanisms into practical design tools. These tools provide scaffolding for novices as they construct an understanding of best practices in design while engaging in real design work.

I begin by introducing Virtual Design Apprenticeship (VDA), a learning model – built on a solid foundation of education principles and theories – that promotes learning of design skills via overlay design tools. In VDA, when an individual needs to learn a new design skill or paradigm she is provided accessible, concrete examples that have been annotated with design rationale. These annotations make expert thinking visible and allow the novice to immediately use, and gradually understand, new best practices. By combining abstract rationale with concrete design instances, annotated artifacts become more useful than either could be alone. I describe the essential components of the VDA framework: annotated design artifacts, a repository of carefully chosen annotated examples, and a community of experts and learners. I walk the reader through an example of how VDA scaffolds learners as they move from a novice's understanding of a design space towards that of an expert. Within the context of this example, I present a set of design principles that guide the creation of VDA design tools – user interfaces built to mediate an individual's interactions with the three core VDA components.

While VDA is applicable to most design fields, I narrow the scope of consideration to one particular domain of design by focusing in-depth on the instructional design difficulties that university-level

faculty members face and how the VDA approach can address them. These instructors face precisely the type of design paradigm shift that VDA was developed to ease as they attempt to move away from traditional, lecture-based pedagogical methods and towards more modern, learner-centered techniques.

I engaged with these instructors and a curriculum design research group in a six-year period of contextual inquiry. Findings from this study influenced my formulation of the VDA framework and the design of PACT, a design tool that leverages the learning principle of making thinking visible to assist novices as they transition from concrete to abstract reasoning about curriculum design. The central focus of PACT is the incorporation of annotated references to pedagogical design patterns – abstract representations of best practices in instructional design. I discuss the iterative design and implementation of PACT in detail, highlighting the ways in which it embodies the VDA design principles for promoting learning of instructional design via overlay design tools.

Next, I study the challenges of converting abstract best practices and design patterns into concrete annotations that can be applied directly to content. My solution, the PACT Annotation Schema, is a formal mechanism for generating tags and pattern annotations from freeform pattern text. Formal representations of patterns are far more useful than generic references, both as scaffolds for learning and for structuring user interactions with design artifacts. Using this schema, I have generated the PACT Annotation Library, a collection of 56 tags and 74 pattern annotations based on the work of the Pedagogical Patterns Project. Visual representations of these formal annotations are the centerpiece of PACT's user interface.

The PACT tool was evaluated in two distinct stages. First, I present a formative study conducted with early, prototype versions of the PACT tool. This study examines the utility of PACT for expert curriculum designers and curriculum research groups, using a sample annotation process – and reflection on the outcomes of that process – to demonstrate that my approach is feasible and useful for those groups. I then present a summative user study of the utility of PACT for novice learner-centered curriculum designers. I demonstrate PACT's significant impact on how novice designers learn from expert-generated examples, how they perceive the credibility of those examples, and the quality of curriculum designs those novices can produce. These findings show that the VDA approach to learning works and that the PACT overlay curriculum design tool is a successful realization of VDA's design principles.

Last, I discuss future directions for this work. PACT is a fully developed design tool that can and should be used by curriculum designers as they create new courses and build their own understanding of the principles of learner-centered design. The PACT Annotation Schema is a useful mechanism that can be further improved to allow the generation of more accurate and complete annotations based on design patterns. The PACT Annotation Library should be continuously expanded as new patterns and principles are developed. Finally, the Virtual Design Apprenticeship model for learning is a robust and highly-principled approach to integrating design learning and design practice. It is applicable across a wide range of design domains and can help promote learning of design skills in them all.

To the memories of my grandmothers:
Dorothy May Twomey Carle, my model of strength and wisdom
Marjorie Marie Moore Brownfield, my model of love and kindness

Contents

Acknowledgements

I would like to begin by thanking my committee members. Thank you to my advisor, John Canny, for your consistent guidance, for giving me free rein to make my graduate experience my own, and for having the vision and drive to create the Berkeley Institute of Design, my beloved home of the past seven years. To Mike Clancy, thank you for always being there to support my research efforts and for leading the UC-WISE group in its crucial mission to improve computer science education. To Marcia Linn, thank you for continuously expanding my horizons and challenging my understanding of human learning and what it means to research that process. I could not have completed this work without the input of all three of you, and I will always be grateful that I had the opportunity to work with such brilliant minds.

Thank you to my parents, Bill and Debbie Carle, and my sister, Caitlin, for everything. There is no reasonable way to summarize your contributions to my world within this text. In short: I could not have started my grand adventure at Berkeley without your support, I would not have persisted through the ups and downs of grad school without your encouragement, and I may well not have finished this dissertation without knowing that you would love me even if I failed. Thank you.

To my partner Kimberly van der Elst: you are the spark that reignited my desire to finish my work at Berkeley and to move on to new challenges. You reminded me of who I am and helped me become myself again. Before I met you, it was me against the world. Now it is us against the world, and I couldn't be happier. I love you; thank you.

Thanks to my labmates and friends at the Berkeley Institute of Design. I can't imagine a better place to work or a better group of people to work with. While everyone that has ever sat in BiD has been significant in my life and career in one way or another, I cannot pass up this opportunity to thank a few of you individually. Thank you to Ana Ramírez Chang and Tye Rattenbury for helping me settle into the life of a grad student and for welcoming me to BiD with open arms. Scott Lederer convinced me to stay at Berkeley and in HCI after a rough first year, whether he meant to or not. Jono Hey, Dave Nguyen, and Jeremy Risner made BiD the place that I love and were a joy to spend time with. Matt Kam, Jeff Heer, and Jingtao Wang were ideal examples of what it means to be an HCI researcher. My good friends David Sun, Wes Willett, Lora Oehlberg, Drew Fisher, Anuj Tewari, Celeste Roschuni, and Kim Lau made my time at Berkeley wonderful – I want you all to know that I blame you for making BiD a place that I was in no hurry to leave. Finally, the new crop of design researchers that arrived at BiD as I was making my plans to depart were a breath of fresh air and rejuvenated my spirit. For that, thank you to Valkyrie Savage, Shiry Ginosar, and Mark Fuge.

Thank you to the UC-WISE group for putting up with me for so many years and for turning what should have been a quick contextual inquiry into the definitive partnership of my Berkeley career. In particular, thank you to Nate Titterton for your friendship and advice. No one at Berkeley was more frank or more correct than you in describing the graduate experience or how to get through it.

Thank you to Dan Garcia for putting your faith in me by giving me significant teaching responsibilities in my first year at Berkeley, making me feel like I had a place and role within the department, and your valuable contributions as a member of my qualifying exam committee. Thank you to Dan Greenstein for giving me an opportunity to look behind the curtain and see how the UC system works while gaining valuable experience. Thank you to Lora Oehlberg for collaborating with me on many efforts over the years and for your lovely artwork that appears in Chapter 2 of this dissertation. And a special thanks to everyone that participated in my user studies – this dissertation would not have been possible if you hadn't have looked at my emails and said, "Why not?"

Thank you to Marjorie Skubic, Sam Blisard, and Bob Luke for giving me my start as a researcher during my freshman year at the University of Missouri – Columbia. I could not have made it to where I did without the push that you gave me. I would also like to acknowledge the Discovery Fellowship Program of the University of Missouri Honors College and those responsible for its creation and success, Stuart Palonsky and Jonnel Clothier. This program provided the financial and administrative support that made it possible for me to get involved in undergraduate research starting from day one of my time at Mizzou – an opportunity that would shape the next eleven years of my life and beyond. It was truly an honor to be among the first generation of Discovery Fellows.

To my old friends Ande Bennett and Pat Hansen, thank you for taking those first steps into the world of programming and computer science with me so long ago. You challenged me to learn and adapt in ways that no formal education ever could have – Programming Gods for life. Thank you also to Michele Cole, my high school programming teacher. You had an under-appreciated job teaching a crucial subject in an overly-bureaucratic school. Thank you for bending the rules to create an environment in which Ande, Pat, and I could learn so very much in so very little time.

Thank you to all my friends from Berkeley and Columbia who have been my support network over the years: My dear friend Rachel Townsend, who was with me through the good and bad of this long, strange trip. Bela Stepanova, who held me up during my awkward transition from undergraduate life to graduate life and lovingly dragged me (kicking and screaming) through the worst semester of my academic career. Thomas Topi, who has never let the distance between Kansas City and Berkeley put a dent in our friendship. Will Dimmit, who was always there to help solve problems, both engineering and otherwise. Whitney Berard, who pushed me to stick with grad school and to see this through to the end. Kevin Smith, Danielle Koonce, and Gretchen Maune, who were my family in Columbia and always will be.

I would also like to acknowledge the sponsoring organizations that have funded my work: the National Science Foundation's CPATH CB program, Microsoft Corporation, and the various donors who support UC Berkeley's EECS Department and the Berkeley Institute of Design.

Finally, thank you to the city and people of Berkeley, CA. My graduate experience would not have been what it was without your hills, your Bay, your trees, the characters of your streets, your drum circles, your half-hearted mugging attempts, your coffee shops, your restaurants, your lofty idealism, your crushing realities, and everything else that makes Berkeley so very Berkeley.

Chapter 1

Introduction

In this dissertation, I study the challenges of promoting design learning within the context of design tools. I propose a general approach and framework, called Virtual Design Apprenticeship, for addressing these challenges. I then narrow the scope of consideration to one particular field of design, focusing in-depth on the instructional design difficulties that university-level faculty members face when attempting to move away from traditional, lecture-based pedagogical methods and towards more modern, learner-centered techniques. I describe a realization of the Virtual Design Apprenticeship approach targeted at these individuals: PACT, a fully-developed curriculum design overlay tool that is intended to both teach about best practices in learner-centered instruction and directly influence curriculum designs as they are being created. I discuss the iterative design and evolution of this system, focusing on how it was crafted to help novice instructional designers *learn from well-designed examples* by exposing the *design principles* and *best practices* that went into creating them. I present a formative evaluation of the utility for expert curriculum designers of an early version of this tool and then a summative evaluation of the final version of the tool's effectiveness in helping novice designers deal with the intricacies of learner-centered curriculum design.

1.1 Motivation

Design is famously described as a "wicked" problem (Rittel & Webber, 1973), one in which there is no definitive formulation, no stopping rule, and no immediate or ultimate test of a solution. As a result, design is not only a difficult task to undertake, but also a difficult skill to learn. A central barrier to learning design skills is that, unlike many rote workplace tasks, crucial processes like creativity and deep consideration of prior experience take place in the minds of practitioners, out of sight and unavailable to inspection by learners. Historically, this barrier was overcome in the training of design practitioners (including architects, engineers, and craftsmen) by employing an apprenticeship model. By closely observing and participating in the process of an expert designer, a novice not only discovers how to use the tools of the trade, but also learns to think like an expert.

Within the realm of formal education, this process is typically situated in design studios with a community of students, instructors, and outside experts. Learners work together to build design artifacts which are then critiqued by the instructor and experts. Design is often taught and understood through experiential learning (Beckman & Barry, 2007; Kolb, 1984), scaffolded by iterative cycles of practice, critique, and reflection (Schön, 1987). As designers go from concrete to abstract thinking, and from analytic to creative tasks, they generate and manipulate representations of their design thinking in the form of sketches, prototypes, and documents (Goel, 1995). These design artifacts trace the learner's progress and can be inspected by experts to gauge understanding.

Unfortunately, neither apprenticeship nor studio learning fit into the life of the practicing professional designer. These learning models require serious commitments: dedicated relationships with experts, synchronous collaboration with teachers and peers, and (often) co-location within a design studio or workplace. And, indeed, practicing professional designers must continuously learn new skills in order to remain relevant and employable. We have entered an age where paradigm shifts in design practices occur increasingly more frequently. Examples include cloud computing for software engineers, sustainable design for new product developers, mobile application design for user experience designers, and learner-centered design for curriculum developers. Each of these emergent practices requires a substantially new way of thinking about design problems.

As a professional designer's skill set ages she must resume her training or risk becoming obsolete herself. Most companies cannot afford to provide continuing training for all of their employees and, despite cheap online options, many individuals face a direct conflict between time spent learning and time spent earning. How will the practicing designer keep up with emerging paradigms when she no longer has physical access to experts in an educational setting?

In this dissertation, I study an approach to answering this question. The key innovation in my solution is the integration of learning mechanisms into practical design tools. These tools provide scaffolding for novices as they construct an understanding of best practices while engaging in real design work. In this model, when an individual needs to learn a new design skill or paradigm she is provided accessible concrete examples that have been annotated with design rationale. This makes expert thinking visible and allows the novice to immediately use, and gradually understand, new best practices. This dissertation describes this learning model in detail and examines the necessary features of the design tools that mediate interactions with annotated examples.

1.2 Thesis

My thesis is that innovative design tools that mediate interactions with annotated design artifacts can help achieve the benefits of apprenticeship and studio-based design learning in a distributed, asynchronous model that fits into the lives of practicing professionals. This is an important step towards promoting design learning and making learning accessible throughout a designer's career.

1.3 Contributions

In this dissertation, I make the following contributions:

- I develop and explain Virtual Design Apprenticeship (VDA), an approach to design learning that revolves around exposing expert design knowledge in the context of authentic design artifacts (Chapter 2). The principal feature of VDA is pattern-annotation, a mechanism that overlays concrete designs with references to design patterns – encapsulated abstractions of best practices in design. This union of design practice and design theory provides a unique opportunity for developing understanding while doing real design work. I provide a set of design principles that any design tool must instantiate as features to realize the benefits of the VDA approach.

- I discuss a significant paradigm shift that is currently taking place within university-level instructional design and show how VDA is an ideal approach to helping instructors keep up with changing practices (Chapter 3).
- I describe the initial design (Chapter 5) and iterative development (Chapter 7) of PACT, a curriculum design tool based on the VDA principles, that I created to help novice learner-centered designers learn from well-designed examples by exposing the design principles and best practices that went into creating them. This demonstrates that the VDA principles can realistically be applied to practical tools and shows the interface features necessary to properly instantiate them.
- I develop and describe a formal mechanism for converting textual descriptions of pedagogical patterns into pedagogical pattern annotations that can be applied directly to real course designs to create pattern-annotated courses (Chapter 6). To demonstrate the utility of this approach, I provide a full library of pattern annotations derived from the public repository of design patterns provided by the Pedagogical Patterns Project.
- I present a formative study of the applications of the initial implementation of PACT for expert curriculum developers and curriculum research groups (Sections 5.4 – 5.5). I find that it has high utility for helping expert designers structure their thoughts, find potential weaknesses in their own designs, and lead curriculum design discussions. I also offer findings from a six-year contextual inquiry process that ran concurrently with the design of PACT (Section 5.6).
- I present a summative user study of the utility of PACT for novice learner-centered curriculum designers (Chapter 8). This study demonstrates that PACT has a positive effect on how novices learn from example curriculum designs, that it impacts novice users' perceptions of examples, and that it helps novices create higher quality curricula.

1.4 Outline

In Chapter 2, I discuss the research from the learning sciences upon which I have based the Virtual Design Apprenticeship framework and then describe in detail the VDA approach to learning, walking the reader through how a learner progresses from novice to expert by interacting with pattern-annotated design artifacts. In Chapter 3, I discuss a significant paradigm shift that is taking place within the field of university-level instructional design – a shift that makes that domain an ideal target for the VDA framework. In Chapter 4, I ground the work at hand within the context of related work on design and learning tools. Chapter 5 describes the initial design of the PACT design tool, including an overview of key functionality, applications of the tool, and a sample course annotation project completed by an expert in lab-centric instruction. In Chapter 6, I present a formal schema for converting textual descriptions of pedagogical design patterns into pattern annotations that can be applied directly to course designs. In Chapter 7, I present a redesigned version of the PACT tool that has been crafted to embody all of the VDA design principles. In Chapter 8, I describe a summative user study of this tool's impact on the experience of novices working to design lab-centric courses. Finally, in Chapter 9 I discuss future work on VDA and PACT beyond this dissertation.

Chapter 2

Virtual Design Apprenticeship: A Pattern-Annotated Approach to Learning

The first contribution of this dissertation is a comprehensive approach to learning that draws upon both research findings from the learning sciences and my own findings from my contextual design process (discussed in Chapter 5). This approach, which I call Virtual Design Apprenticeship (VDA), revolves around making the way that experts think about the curriculum design process visible and accessible to novices. By exposing this rationale in an easy-to-understand format, VDA facilitates a self-guided learning modality in which novices can explore and build understanding at their own pace. The learner uses the expert's thoughts as a source of truth against which they can compare their own ideas on design. Thus, the expert becomes a sort of virtual coach, helping guide the novice towards a richer understanding of the design space.

2.1 Human Learning Background

VDA is built on top of the foundation of research provided over the past century by the learning sciences community. Any method intended to help novices learn about best practices in design must be informed by a broad understanding of how people learn. Further, to address VDA's target audience of practicing designers, I must deeply consider the motivations of adult learners and the strategies that work best for teaching active professionals. In this section, I review relevant findings from the educational psychology community that have influenced the VDA framework.

2.1.1 Zone of Proximal Development & Instructional Scaffolding

The fundamental finding from the past century of education research is that learning is an inherently social process. Learners progress through stages of understanding by working with experts and peers or with artifacts created by experts. Lev Vygotsky modeled this process with the zone of proximal development (ZPD, Figure 2.1), described as "the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance, or in collaboration with more capable peers." (Vygotskii & Cole, 1978) In my context, we are interested in what curricula a novice designer can create alone and what they can create with the guidance of an expert. The

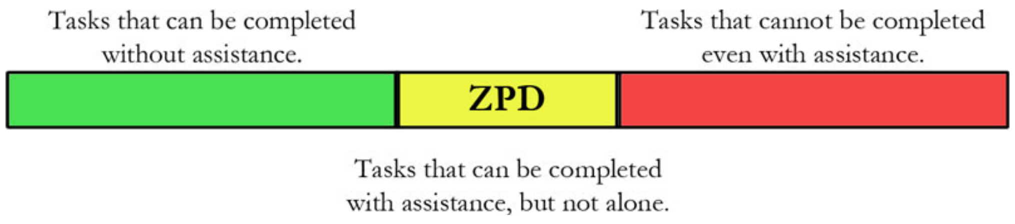

Tasks that can be completed without assistance.
Tasks that cannot be completed even with assistance.
ZPD
Tasks that can be completed with assistance, but not alone.

Figure 2.1: Vygotsky's Zone of Proximal Development

changing nature of the ZPD over time can be used to measure an individual's learning – if he is learning, then tasks that were once only possible with assistance will become possible with none. In contrast, if the ZPD is not moving over the course of a teaching experience, then something is amiss. Most likely the learner is either not being sufficiently challenged (presented tasks can be done without assistance) or is being over challenged (presented tasks cannot be done even with assistance). Managing the learner's progress through the ZPD is crucial to promoting learning and is the main goal of VDA.

Instructional scaffolding (Wood, Bruner, & Ross, 1976) is providing assistance to learners that helps them achieve what they could not alone. In essence, it is the support that helps a learner move through the ZPD and towards autonomous work. Good scaffolding reveals just enough information to the learner to continuously challenge their conceptual model of a problem, but not so much information that they feel overwhelmed. Learners interact with scaffolding in an iterative manner, gradually moving from a novice's view of a problem towards that of an expert. More recent work on the subject (Saye & Brush, 2002) delineated soft and hard scaffolds, with the former being flexible and dynamic and the latter static and planned in advance. This distinction is of note to my work, as the scaffolding that can be encoded in a software application is, by its nature, hard scaffolding. Scaffolding that can be provided interactively by community members is the more valuable soft scaffolding. VDA provides both types of scaffolding via a design tool that mediates interactions with annotated design artifacts created by experts.

2.1.2 Legitimate Peripheral Participation and Andragogy Theory

The social contexts in which learners best construct new knowledge have been extensively studied. Learners (especially adults) progress most quickly while working on authentic problems in a community of practice comprised of peers and experts (Lave & Wenger, 1991). Ideally, this community engages in real work together by actively collaborating and conversing about common struggles. Novices participate in these efforts to the extent that they can while learning from more senior community members through socialization, visualization, and imitation; a process Jean Lave & Etienne Wenger refer to as legitimate peripheral participation (1991). The work of experts serves as the instructional scaffolding needed for the novice to learn and begin participating in the community more centrally. VDA enables this progression by exposing the design rationale of an expert within the context of that expert's work.

Malcolm Knowles' theory of andragogy (M. Knowles, 1984; M. S. Knowles & Bard, 1984) describes how adult education should be structured and how it differs from traditional classroom education. He finds, most pertinently, that adult learning must be largely self-guided, self-structured, and be centered on practical experiences. Adult learners are drawn to materials that have immediate relevance to their careers and are more compelled by task-centered educational framing than by content-oriented models. Adults tend to be motivated by internal goals and aspirations; they are repelled somewhat by overly aggressive external motivators. VDA is built upon these principles, motivating learners through authentic design work that can immediately be put to real use.

2.1.3 Cognitive Apprenticeship & "Making Thinking Visible"

Allan Collins describes the cognitive apprenticeship model (Collins, Brown, & Holum, 1991), a framework for education that centers on skillful masters explicitly making their thinking visible to apprentice learners. Masters must be compelled to work actively towards this goal, as they otherwise have trouble expanding upon the tacit processes that are needed to successfully complete complex tasks. This is often accomplished by having the expert "think aloud" while considering real-world problems. VDA leverages this learning paradigm in a distributed, digital learning model via the use of annotated design artifacts.

2.1.4 Progressing from the Concrete to the Abstract

Recognition of the importance of a progression from concrete to abstract thought dates back at least to the philosophies of Immanuel Kant. Jean Piaget grounded this progression in empirical results from childhood development while formalizing his four stages of development, which generally move from more concrete to more abstract operations (Piaget, 1970). Essentially, learners often find it much easier to deal with real-world (often literally physical) concepts before moving into abstractions of the concepts. An example taken from Staub and Stern (1997) illustrates the concept well. Imagine an elementary mathematics student asked to consider the following problems: summing two piles of 5 and 12 marbles sitting in front of them, reading a story about 5 and 12 trout and being asked to explain how many total trout there are, "5 + 12 = ?", and "$a + b = c$", where $a, b,$ and c represent sets of numbers such as integers. It is easy to see how the more physical, concrete examples are easier to understand while the more abstract concepts require deeper consideration. VDA scaffolds this transition, helping learners build understanding of complex, abstract concepts through mediated interactions with concrete examples.

2.2 The VDA Components

Ultimately, the goal of VDA is to capture the flavor of cognitive apprenticeship in an approachable and accessible model for lifelong learning. VDA helps professional designers keep current with emerging paradigms in their field by teaching them in the context of their own authentic design work – avoiding commitments to organized courses or independent training that is divorced from their professional practice. VDA is comprised of three key components:

1. **Annotated design artifacts.**
2. **A repository of carefully chosen example designs.**
3. **A community of experts and peer learners.**

Interactions with these components are mediated by design tools that expose learners to expert thinking and best practices while they engage in practical design work, allowing incremental improvements in understanding of the design space over time. The iterative design and implementation of one such tool, PACT, which is aimed at helping university-level instructional designers, is the focus of the remaining chapters in this dissertation.

2.2.1 Annotated Design Artifacts

The heart of the VDA approach is the design rationale-annotated design artifact—a concrete representation of a design with annotations that describe the intent behind why the design was created the way it was. A design concept is codified both in the organization of the artifact's content and in annotations created explicitly by expert designers. Through annotated design artifacts, experts make explicit the tacit knowledge that they have developed through years of experience. Exposure to this tacit knowledge helps learners structure their thoughts on the design space and best practices. Thus, annotated artifacts pull in desirable benefits from the learning sciences:

- They enable cognitive apprenticeship in a distributed, asynchronous manner.
- They provide the necessary instructional scaffolds to help a learner continuously progress through the zone of proximal development.
- They make abstract concepts concrete, but provide a route for the learner to progress back from the concrete to the abstract.

I have identified two types of annotations that should be applied to design artifacts to enable cognitive apprenticeship: *tags* and *structural annotations*. *Tags* allow the user to label each design element and note the purpose or role of that element within the overall artifact. They also may describe why a particular design element was chosen rather than another candidate solution. The artifact's collection of tags becomes a vocabulary that represents all the elements of a design and what each element is intended to do.

Structural annotations describe relationships between components of the design and are expressive at a variety of scopes within an artifact. They go beyond individual decisions and can frame the way that a learner looks at an entire design space. Such annotations describe phenomena at the level of the entire design all the way down to its individual components.

A well-annotated artifact is self-explanatory. The annotations describe what each piece of a design is, what it does, how it interacts with the rest of the design, and why it was chosen over other competing solutions. In this regard, annotated artifacts act as concrete versions of case studies (M. C. Linn & Clancy, 1992). But where creating good case studies requires considerable pedagogical expertise, annotated artifacts seek to expose design knowledge as a natural consequence of real design work.

2.2.2 Repository of Carefully Chosen Examples

While a single well-annotated design artifact provides insight into the thought process of an individual expert, a repository of many carefully chosen annotated artifacts can reveal the entire landscape of design thinking within a domain. By exploring a number of divergent – but equally valid – solutions to similar problems, a learner can build her own understanding of the design space and construct a model of how and why emergent paradigms are being incorporated into authentic designs. This gives agency to the learner and allows her to proceed with building understanding in a self-guided, self-structured manner—both key factors for the success of adult learners per Knowles' andragogy theory.

The annotated design repository should be curated and presented in a manner that improves the learner's probability of success without reducing her sense of agency. A learner that is first being introduced to a new design paradigm should be exposed to straightforward examples that clearly illustrate that paradigm's most important aspects. As the learner progresses, more complex and divergent examples can be revealed; examples that, for instance, include more complicated, controversial, or opaque instantiations of the new design methods. Thus, the annotated designs become effective instructional scaffolds and help lead the learner from a novice view of the design space towards that of an expert.

2.2.3 Community of Experts and Learners

The structure of annotated design artifacts and the examples in a well-curated repository provide the fundamental blocks for constructing knowledge of emergent design paradigms. Weaving these pieces together into deep understanding, however, requires guidance from experts and collaboration with other learners. In a traditional studio environment, these interactions happen with a dedicated cohort of co-located peers, instructors, and outside experts. In VDA, this environment is simulated with a distributed, asynchronous online community of practice. This is this community that allows learners to develop via legitimate peripheral participation and that provides the experts that drive the cognitive apprenticeship model.

The success of this transition is only possible due to the presence of annotated design artifacts. By exposing the thought process of all members of the community (learners and experts alike), the exchange of annotated artifacts mediates discussions about design in both directions. Thus, *VDA makes apprenticeship possible without the normal requirement that experts and learners coordinate synchronously*. Experts annotate their designs to make their tacit knowledge explicit and available for learners to consume at their own pace. Learners annotate their work to expose misconceptions or errors that can be recognized and corrected by experts during asynchronous design critiques.

Within this community of practice, *VDA distributes teaching in a manner that achieves the apprenticeship effect, but does not require fixed master-apprentice relationships.* Via annotated artifacts, a learner can consume the knowledge of many experts and receive feedback from many other community members. Each of these interactions is grounded in a concrete representation of design thinking that is available for everyone to inspect. While critiquing a design, an expert can use the annotations to see how the learner's thought process has evolved over time, what expert-created designs inspired their work, and what advice has been given by other members of the community. This makes it possible for each member of the community to jump in and out of the discussion smoothly, allowing participation at a level that is comfortable for the individual. By lowering the barrier to entry, VDA facilitates expert participation and helps learners approach the community with less fear of a serious commitment.

2.3 How VDA Scaffolds Learning Transitions: Adopters, Adapters, Combiners, & Creators

To better understand the VDA model and how it can be realized for a design domain, it is useful to consider the progression of an individual learner from the point that she decides to investigate a new design paradigm through the point at which she has built enough knowledge to begin working independently. Based on published work in design education (e.g. by Schön (1987) and Goel (1995))

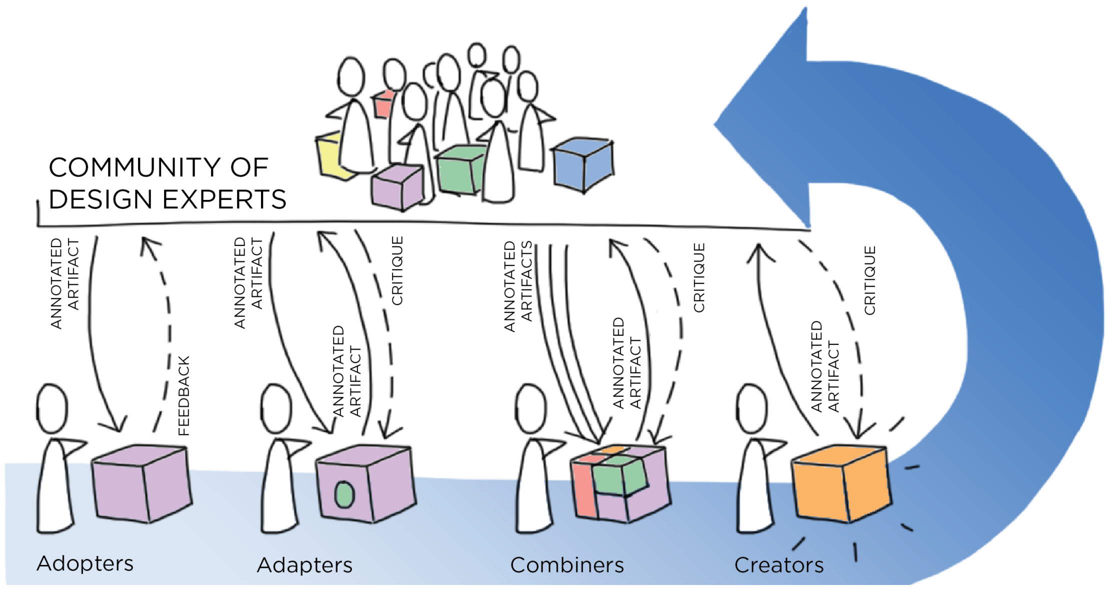


Figure 2.2: The VDA framework describes how learners develop design skills, as scaffolded by interactions with expert-annotated design artifacts from a curated repository. The four stages of learners are a) adopters, who utilize experts' artifacts; b) adapters, who slightly modify expert-annotated artifacts; c) combiners, who transform multiple expert-annotated artifacts into a new design; d) creators, who build entirely new annotated artifacts. Learners interact with the design community through feedback from the adopters on the performance of experts' artifacts and critique from experts on the learner's artifacts. Interactions with artifacts are mediated by VDA design tools.

and my own experiences, I will use the following abstraction to describe the evolution of learners using a VDA tool. In this model, the learner moves through four phases of design understanding (Figure 2.2): (1) *adopting* others' designs; (2) *adapting* others' designs to fit new design contexts; (3) *combining* new designs from pieces of existing designs; and (4) *creating* entirely new, novel designs. The learning phases occur concurrently with professional practice and may span several years. They are neither discrete nor strictly ordered – an individual may move back and forth between them over time, incorporating earlier strategies in later phases of learning as the situation demands.

In the following sub-sections I will describe how learners interact with annotated design artifacts, a repository of carefully chosen examples, and a community of experts and peers to scaffold progress through each phase of the learning progression. Throughout this discussion, I will highlight design principles (summarized in Table 2.1) that guide the creation of VDA design tools – interfaces built to mediate an individual's interactions with the three core components of VDA. The first of these principles is that VDA design tools must **enable design artifact creation and annotation**, a requisite step for all the other interactions described below.

2.3.1 Supporting Learning Phase 1: Adoption

A learner in the *adoption* phase is primarily interested in finding relevant designs and employing them directly. In order to support adopters, VDA design tools must **facilitate sharing annotated design artifacts and finding relevant designs**. Adopters must have easy access to the repository of expert-generated content and should only be exposed to the most complete and easily understood examples. VDA design tools should also **make design artifacts usable in the real world**, helping the designer shift from concept to production (publication, fabrication, etc., depending upon the

1. Enable design artifact creation and annotation.
2. Facilitate sharing annotated design artifacts and finding relevant designs.
3. Make design artifacts usable in the real world.
4. Make annotated artifact exploration and comprehension easy.
5. Encourage design experimentation, but preserve expert design intent.
6. Structure community discussions around the annotated design artifact.
7. Use annotations as templates for new designs.
8. Link annotations directly to further reading.
9. Encourage learners to provide performance feedback to experts.

Table 2.1: The VDA principles for promoting design learning within design tools.

design domain). The goal is to make the tools as useful to the professional designer as possible while showing them the route (via design annotations) to deeper learning that is available when they are ready.

2.3.2 Supporting Learning Phase 2: Adaptation

As design artifacts are built to solve a specific problem in a specific context, applying a good design to a new context (while maintaining its quality) requires some customization. This need to change an expert-created design forces the learner to transition to the second phase of design learning: *adaptation*.

Successfully adapting a design requires a fledgling understanding of how the design's elements work together. In VDA, the adapter examines the construction of an expert-created artifact (still selected from the most complete and easily understood examples in the repository) to begin building this conceptual model. The thought process of the expert is made visible through the annotations – at this point adapters are likely to deeply investigate the simple tag annotations of the design while merely skimming the more complex structural annotations.

To enable these tasks, VDA design tools must **make annotated artifact exploration and comprehension easy.** It should be possible to quickly move between examining the details of a design and taking in the big picture of the artifact. Annotations should be made available in a succinct and salient way.

Adapters are just beginning their move through the ZPD; they are not yet capable of building radically new design artifacts, but they can experiment with small details of an existing design. Scaffolding in the form of prompt critiques is crucial for maintaining learning progress; realizing that a design change has created an error is the first step in correcting and understanding that error.

VDA structures both personal and expert critiques. When manipulating an annotated section of the design, the adapter needs to continuously verify that the artifact's annotations remain accurate throughout their adaptation. If they do this, the principled intent of the expert designer is maintained. Self-monitoring adherence to the annotations encourages meta-cognitive reflection on the act of design, which in turn promotes learning (Flavell, 1979). At the community level, the learner can share her changes with experts for critique. Annotations make this process much easier, as experts can quickly spot mismatches of annotation and implementation. Essentially, tools supporting VDA should **encourage design experimentation, but preserve expert design intent**.

2.3.3 Supporting Learning Phase 3: Combination

Making small changes to an existing design will only take the learner so far. As she builds a solid understanding of design practices and considerations within her domain, she will find her ideas expanding beyond the bounds of any single existing design. Not yet ready to create an entirely novel design on her own, she will begin assembling new design artifacts from elements of many existing designs. At this point, the learner has entered the third phase of design learning: *combination*.

Combiners must develop a more robust and broad understanding of expert-created design artifacts than was required for adaptation. In this phase, they delve deeply into the structural annotations of artifacts, mastering the intentions of the original designers and building a clear mental model of how the elements of each design are interrelated and interdependent. The combiner gathers many divergent examples of all levels of complexity from the artifact repository and weighs the pros and cons of pieces of each. Structural annotations assist in this process by clearly delineating units of the design and describing why they should remain intact.

Combinations of design ideas are inherently more prone to error than simple adaptations; critiques must be more robust, more frequent, and more specific to prevent the learning process from stalling. Thus, learners at this stage will communicate more consistently with their community of peers and experts. It is of paramount importance that all participants in this conversation be able to refer to specific elements of design artifacts quickly and easily. In order to mediate this discussion, VDA design tools must **structure community discussions around the annotated design artifact**. This provides a solid reference point to ground conversations about design that could otherwise easily venture too far into the abstract to be useful for the learner.

2.3.4 Supporting Learning Phase 4: Creation

Once the learner has developed a rich understanding of her design domain she will feel constrained by working within others' artifacts and structures – she now builds entirely novel artifacts. This transition need not be made in one dramatic leap. More commonly, a combiner begins building new design elements to augment artifacts from the repository, taking gradual steps into the fourth phase of design learning: *creation*.

To make this process tractable, VDA aims for incremental moves away from using existing designs and towards building new ones. The first step is to help users adopt the *structure* of expert-created designs without using any of the *concrete elements* of that design. Burgeoning creators can use existing designs as templates for new designs. A structural annotation provides the unit/scope of a template. The semantics of that annotation combine with the meaning of tags within the template structure to provide a concrete guideline for how new design elements should be created. Any new design that conforms to the constraints implied by the annotations will also embody the principles intended by the original designer. Because of this, VDA design tools should **use annotations as templates for new designs**.

The next step in moving towards independent creation is developing an understanding of *why* certain practices are considered optimal in a design space and why others are not. Truly novel designs can break new ground in defining best practices and generate new annotations and patterns that describe them. But making this leap requires expert-level knowledge. The expert creator has read the theoretical background and considered the relevant empirical results that support or rebuke specific design practices. To support this knowledge acquisition, VDA design tools must **link annotations directly to further reading**.

2.4 Expert Participation and Incentives

The VDA model relies heavily on expert participation in the community of practice. Busy designers are, however, unlikely to take time out from their schedules to help anonymous learners without some incentive. Therefore, in addition to enabling artifact annotation, the annotation process and the completed annotated artifact should be directly useful to the expert designer. VDA design tools must be compelling to their target expert community. In my experience, utility to experts most often comes in the form of assisting with two tasks: *reflection* and *dissemination*.

During the annotation of a completed design artifact, the expert has an opportunity to reflect (Schön, 1987) upon their design decisions in a structured manner. In doing so, they can find problems with their design that were overlooked in their initial construction of the artifact. For example, the expert may identify anomalous portions of a design that do not exemplify a design pattern that is ubiquitous throughout the rest of the artifact, as seen in my formative user studies described in Section 5.5.

Artifact annotation can also help with design dissemination and proselytizing the experts' particular viewpoint or preferred methodology. Clearly demonstrating these viewpoints in the form of annotated artifacts as a resource for others can spread new design paradigms or methodologies to learners who are just beginning to form their own ideas on the art of design. The process of creating these annotations helps clarify the expert's opinions to herself, allowing her to build on them in future artifacts.

Dissemination can be further aided by using the community of practice itself to gather evidence in support or opposition of particular design ideas. If every learner that uses an expert's design ideas provides feedback on what did and did not work as advertised, the expert will be able to quickly verify the strength of their design. This can be a powerful incentive and resource to experts interested in establishing the validity of their design paradigm for publication. Therefore, design tools supporting VDA should **encourage learners to provide performance feedback to experts**.

Chapter 3

Applying VDA to University-Level Instructional Design

To investigate the utility of the VDA framework in a concrete way, I decided to narrow the scope of my inquiry to a specific design domain. I chose instructional design at the university level to be the focus of my design and implementation efforts for the following reasons:

1. Best practices in instructional design are rapidly changing in response to an evolving understanding of how students learn. This paradigm shift in professional design practice exemplifies the problem I am trying to solve with VDA.
2. Significant research effort has been invested in developing a pattern language that describes these new best practices. This language can be converted into a collection of annotations as described in Chapter 6.
3. The Lab-Centric Instruction teaching method, invented and researched by the UC-WISE group of UC Berkeley's Computer Science Division, is an excellent platform on which to test the VDA principles and the utility of pattern-annotation-augmented design tools.

My work in the realm of instructional design includes a full implementation of a design tool based on the VDA principles. This system, called PACT, is described in Chapters 5 and 7.

3.1 Paradigm Shifts in Instructional Design

Theory-driven best practices from learning research have diverged from the teaching style practiced in university classrooms. Over the past forty years, a wide variety of non-traditional instructional techniques have been developed and validated. The most compelling of these are based on the notion of learner-centered instruction. The unifying theme of this style of teaching is systematic focus on students engaged in learning as an active process. Due to strong empirical results and solid grounding in the theoretical frameworks of psychology and education, learner-centered instruction is the emergent recommendation of educational researchers and has been promoted in essentially every policy document by learning experts, including the influential National Research Council Report by Bransford, Brown, and Cocking (2000).

These methods have shown great promise, producing significant learning gains over conventional practices and often leading to dramatically improved student outcomes. An early canonical example of such a change is Tutored Videotape Instruction, introduced in the 1970s (Gibbons, Kincheloe, & Down, 1977). More recently, learning gains have been shown through the use of inquiry-based learning (Laurillard, 1993; Williams & Linn, 2002), collaborative learning (Kramarski & Mevarech, 2003), peer instruction (Crouch & Mazur, 2001), and many other approaches.

Despite the successes of these new pedagogical practices, outdated and deficient pedagogical techniques continue to dominate in university-level teaching environments, both in traditional university instruction and in online course design. The typical university classroom experience has not changed significantly in the past century. College students spend most of their class time in lecture halls, focusing on a single source of information: the words of the instructor. 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. Anecdotally, many computer science instructors in traditional courses acknowledge that "most learning happens in the labs" but still "do the teaching in lectures." While this modality of learning does work well for some students, it appears to not work at all for many, and is certainly sub-optimal for most (Bransford, 2000; Laurillard, 1993).

Concurrently, the discussion of online teaching practices (both within universities and in the popular press) has shifted towards massively open online courses. This enthusiasm has been driven by the successes of Kahn Academy, edX, Coursera, Udacity, etc. in registering large numbers of students. In contrast to the principled pedagogy and careful design of learner-centered courses, these approaches have been criticized for their often shallow and straight-forward adaptation of the traditional lecture model. They typically fail to use established best practices for computer-mediated learning, have very low completion rates, lack scientific evaluation, and exhibit poor student outcomes in the few cases where evaluations have occurred.

Given the potential benefits, why has the university community been slow to embrace new pedagogical practices? There are several barriers to adoption, including technical limitations, conflicting demands on faculty time, and administrative expectations. At the core of these problems lies the complexity of creating and implementing a learner-centered course. Most professors simply don't have the time or resources to learn about these techniques and develop curricula that incorporate them. A secondary issue is that most instructors' conceptual model of student learning, at least in engineering and the hard sciences, is one of "filling students with knowledge" – formally known as transmission teaching. It favors the lecture model over holistic learner-centered design. It conflicts with virtually everything that is known about human learning and with studies of best practices, but instructors who have not explored learner-centered design have had no chance to compare the approaches. Instructors therefore cannot design learner centered content with understanding. Instead, learning the principles becomes part of learning the practice of learner-centered design. This creates a double challenge, but also an opportunity to improve both practice and understanding.

3.2 Pedagogical Design Patterns

VDA relies upon the existence of domain-specific libraries of structural annotations and tags that describe the domain's design space. I have developed one such library of annotations (described in detail in Chapter 6) for the domain of learner-centered instructional design. That work was based upon Pedagogical Design Patterns (PDP) (Sharp, Manns, & Eckstein, 2000), a promising mechanism for abstracting and sharing best practices in learner-centered curriculum design. Inspired by

Alexander's seminal *A Pattern Language* (Alexander, Ishikawa, & Silverstein, 1977) and an article by Susan Lilly (1996), the goal of a PDP is to identify an outstanding teaching practice, abstract the generally reusable element of that practice, and disseminate it. PDP descriptions facilitate a common vocabulary for pedagogical research and practice, are accessible to novice designers, and encourage the repurposing and reuse of techniques that are grounded in modern learning theory. Early focus was on teaching object-oriented concepts; subsequent work by Joe Bergin (2000) and others has extended the focus to other areas of computer science instruction and beyond.

In the spirit of Alexander's 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 should be able to determine if the pattern applies to her situation. Ideally, the pattern should be general enough that the instructor can tailor it to her own needs without negatively altering the pedagogical theory.

The members of the Pedagogical Patterns Project have created, tested, and disseminated many patterns of this nature. Two examples of these patterns that demonstrate the range of scope covered by the project are Spiral and Early Warning. The Spiral pattern (Bergin, 2000) is essentially a pattern encoding of Bruner's spiral curriculum (J. Bruner, 1977). It provides course-level advice about topic coverage in a course where there are a large number of concepts that must be mastered together. It suggests covering all of the myriad topics in a course in a low-detail way first, then progressively revealing more particulars of each subject, spiraling in to the most detail-rich content near the end of the course. In this way, the teacher can assume that the student has some knowledge of related course material when covering the details of a topic. Early Warning (Bergin, Eckstein, & Sharp, 2002a) operates over a much smaller scope. It suggests that students be given an opportunity to realize that they are struggling or that they have developed misconceptions earlier rather than later. This allows students to correct the situation while they still have the opportunity to do something about it, rather than on a major exam or project.

3.2.1 Weaknesses of the Pedagogical Patterns Project

In 2003, Helen Sharp and other members of the Pedagogical Patterns Project commented on the progress of the project: "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." (Sharp, Manns, & Eckstein,

2003) This difficulty is confirmed by the fact that, despite their potential, pedagogical patterns have not yet been widely adopted by educators.

It is evident that these difficulties are amplified by the abstract nature of patterns. Given only a pattern description (even a well-written one), it can be very difficult to envision exactly what a successful instantiation of that pattern would look like. Worse, the Pedagogical Patterns Project presents their patterns only in the form of lengthy collections contained within large PDF files. I believe that these text-only descriptions of patterns are too complicated to be useful as learning tools for novices, too inaccessible to be easily appropriated and adapted, and too abstract to be generative while creating new course designs. Further, the process of identifying a pattern that applies to a specific situation is tedious at best. It certainly does not fit into the work flow of the average busy college professor. I see the opportunity for a major contribution to the field by making patterns more concrete, easier to locate, and simpler to appropriate.

In the VDA approach to learning, I seek to remedy these issues and leverage the potential of PDPs by focusing on concrete *pattern-annotated artifacts* – real-world designs that have been marked up with references to patterns.

3.2.2 Scaffolded Knowledge Integration Framework

Marcia Linn's Scaffolded Knowledge Integration (SKI) Framework (M. C. Linn & Eylon, 2011) is a collection of domain independent practices designed to aid students in developing integrated understanding of complex topics. SKI conceptualizes a student's understanding of a topic as an array of interrelated conceptual models. Well-designed learning systems that embody the SKI principles will help a learner develop more sophisticated models and distinguish between accurate, inaccurate, and incomplete models of the topic. At its heart, the SKI Framework prescribes a set of four facets that encourage this process of model building and distinguishing: identifying new goals for learning, making thinking visible, encouraging autonomous learning, and providing social supports.

This work led to the identification of a set of four fundamental processes involved in knowledge integration: (1) eliciting current ideas, (2) adding new ideas, (3) developing criteria for evaluating ideas, and (4) sorting out ideas (M. C. Linn & Eylon, 2006). Further, Linn and Eylon have distilled the state of the art in knowledge integration research to identify 10 instructional design patterns, each of which incorporates the four knowledge integration processes in a unique way. These SKI patterns lack the strict prescriptive structure of the Pedagogical Patterns Project design patterns, but compensate by being more generally applicable to a wide range of instructional settings and by being strikingly well grounded in educational theory.

Illustrate Ideas is an example of a SKI design pattern (M. C. Linn & Eylon, 2011). In this pattern, the four knowledge integration processes are involved by: (1) eliciting the repertoire of ideas for a topic, (2) modeling the process of considering alternatives for a complex problem, (3) identifying emergent or established criteria to distinguish alternatives, and (4) enabling learners to sort out their ideas based on model and criteria. As with the patterns of the Pedagogical Patterns Project, the full text of a SKI pattern fleshes out the details that go on top of this outline and provides examples of successful pattern implementations.

3.3 Lab-Centric Instruction

Lab-Centric Instruction (LCI) (Titterton, Lewis, & Clancy, 2010) is a university-level teaching modality that emphasizes supervised, discovery-based learning approaches by substituting in-lab time in place of lecture time. The most prominent instantiation of the LCI model is the UC-WISE project (M. Clancy, Titterton, Ryan, Slotta, & Linn, 2003), a computer science education research effort spearheaded by UC Berkeley's Michael Clancy. In a LCI-based course a student typically spends six hours in lab and one hour in lecture each week, in contrast to the more standard two hours of lab, three hours of lecture, and one hour of recitation. Attendance in LCI labs is mandatory and the labs are always supervised by at least one member of the course staff.

The key innovation of LCI is a focus on learner-centered content delivery in a supervised environment. Students progress at their own pace through active materials (quizzes, discussions, exercises, guided reflections, etc.) while under the watchful eye of a member of the course staff. This staff member is charged with quickly intervening as soon as a student begins to struggle so that misconceptions can be corrected before they have a chance to do any harm. This approach has been shown to generally help students who struggle in lecture-based courses and is believed to provide improved opportunities for traditionally underrepresented students in computer science (Titterton, Lewis, & Clancy, 2010). While the benefits of lab-centric instruction are substantial, like most shifts in design paradigm there are significant barriers to adoption (Titterton & Clancy, 2007).

I have chosen to focus on LCI as a model instructional modality for the work described in this dissertation. A number of factors have contributed to this decision. First, LCI's strong track record and theoretical grounding make it a sound choice for an instructor looking to take their first steps into learner-centered pedagogy. Second, UC-WISE courses (and fractions thereof) are relatively easily shared with a community and can easily be appropriated into new course designs. Finally, the activities that make up a lab-centric course are a strong fit for the annotation-based design approach of VDA due to their modularity and granularity.

Chapter 4

Related Work

I situate the work at hand in the context of literature related to creating design tools, existing representations of learning design, tools that support pattern modeling, design rationale capture and reuse systems, technologies created to help deliver robust learner-centered instruction, and professional development communities for educators.

4.1 Design Tools

The HCI community has created many design tools for distributed (Klemmer, Newman, Farrell, Bilezikjian, & Landay, 2001) and co-located (Arias, Eden, Fischer, Gorman, & Scharff, 2000; Hartmann, Morris, Benko, & Wilson, 2010) collaborative settings. These tools have also addressed the issue of transitioning between different forms of representation (Gross, 1996) and experimentation (Hartmann et al., 2006). When augmented with additional data (for example, in Designers' Outpost design history), this was primarily to support active design tasks (Klemmer, Thomsen, Phelps-Goodman, Lee, & Landay, 2002) rather than to foster learning. Where design tools have considered learning (e.g. Bjoern Hartmann's work (Hartmann, MacDougall, Brandt, & Klemmer, 2010) or Adobe's Flex Case Studies) they have tackled only portions of the problem. The VDA approach provides much needed context for these efforts – a holistic framing of learning within design tools.

James Landay and his collaborators' works on informal and sketch-based interfaces for design support are notably relevant to the PACT tool's design. These include SILK (Landay & Myers, 1995), a support tool for application design; DENIM (Lin, Newman, Hong, & Landay, 2000), a tool for sketching out the high-level layout of web sites; and Damask (Lin & Landay, 2002), a sketch-based tool for developing application for multiple devices. Damask is of particular interest as it was designed to leverage design patterns to support design tasks. However, Damask focuses primarily on very small-scale, concrete best practices (those that, in the patterns literature, are often termed idioms rather than patterns) that directly prescribe the design process. This differs considerably from the generative approach to patterns that I pursue in my work. The major lesson to take from these projects is their focus on informal representations of knowledge. These representations have been shown to encourage creativity and rapid iteration over designed artifacts – goals that I support in PACT.

4.2 Representations of Learning Design

While few researchers have built systems with the express intention of supporting the design of curriculum, a number of more general design tools have been appropriated in the pursuit of learning design. Most prominent of these are word processors (e.g. Microsoft Word) and presentation tools (e.g. Microsoft PowerPoint). In fact, in my experience, the vast majority of curriculum developed at the university level (especially in the sciences and engineering) has been authored and disseminated primarily in PowerPoint. Recently, instructors have begun using web pages, learning management systems (e.g. Moodle), and wikis to organize their thoughts. These methods all allow for certain amounts of collaboration and coordination, but provide no explicit support for the process of building curricula. Further, these tools provide no real mechanisms for identifying and noting design rationale.

The work of Goodyear et al. (2005) describes the use of structured texts and templates to focus the curriculum design process around design patterns. However, it does not prescribe any particular approach to developing courses around patterns, nor does it suggest a particular design-support strategy. Similarly, concept mapping tools have been used in the educational domain as a means of organizing topics and high-level learning goals (Inglis & Bradley, 2012). However, this work has focused primarily on organizing whole programs of learning, as opposed to the specific details of a course's design. Further, such tools still do not specifically support the design process of building curricula – they are essentially tools for representing any types of interrelated concepts and thoughts. I seek to provide far more precise support for the curriculum design process and for generating appropriate curricula based on examples and patterns.

4.3 Pattern Modeling

Pattern-instance relationship descriptions have been explored in the context of exiting development tools. For example, in Unified Modeling Language (UML) diagrams such groupings are represented with parameterized collaborations (Booch, Rumbaugh, & Jacobson, 1999). A similar user experience can be achieved with logical collections in a concept map (Trochim, 1989) and other general-purpose visual organizational tools (e.g. Microsoft Visio).

It is my finding through my user-centered design process that the affordances of these tools are too general and too far removed from curriculum developers' own experiences to be useful for learner-centered curriculum development. The key to making these features useful and usable is to integrate them into the design process by overlaying them on real design tool. Further, the pattern annotations in PACT are explicit design scaffolds intended to assist in the creation of new content, as opposed to mere identifiers of related items.

4.4 Design Rationale Systems

A number of researchers have attempted to capture information about the design decision making process with design rationale capture systems. MacLean et al. first formalized the concept in their 1989 paper, Design Rationale: The Argument Behind the Artifact (MacLean, Young, & Moran, 1989). They envisioned a formal structure for describing why a designer made the design decisions that they did and, importantly, why they did not choose other competing solutions. Subsequently, design rationale systems have been built for a variety of disciplines including architecture and construction, mechanical engineering, and software and user-interface design (Regli, Hu, Atwood, & Sun, 2000). The intent of these systems is to assist professional designers in structuring design problems, exploring additional options, and sharing their work with collaborators.

Such design reuse and rationale systems struggle to achieve success – Horner and Atwood (Horner & Atwood, 2006) identify the essential barriers to use as cognitive, capture, retrieval, usage, and organizational. However, while Horner and Atwood argue that the ultimate purpose of design rationale systems is to improve the quality of designs, I argue that communication of design rationale should be used to *improve the quality of the designers themselves*. This reframing of the utility of design rationale can provide valuable context for contemporary work on repositioning design rationale, such as that of Branham, Harrison, and McCrickard (2010).

4.5 Educational & Instructional Technology

Many designers and researchers have developed educational technologies to ease the delivery of learner-centered content and improve the student experience. Advanced learning management systems (e.g. Moodle, WISE (M. C. Linn, Clark, & Slotta, 2003), UC-WISE (M. Clancy, Titterton, Ryan, Slotta, & Linn, 2003)) structure course materials in ways that are easily digested and (in the best cases) encourage reflection and other meta-cognitive processes. Other platforms, such as Livenotes (Kam et al., 2005), seek to bring the affordances of small-group discussions and tutoring into the lecture environment. Commercial peer-response systems that attempt to largely automate the process of running peer instruction sessions in the classroom are now widely available and have been used with some success (Milner-Bolotin, 2004). In short, the state of the art in instructional technology is catching up with modern recommendations from pedagogical specialists and educational researchers.

These delivery platforms have shown remarkable improvements in learning when introduced into the classroom in appropriate ways and *used with appropriate content*. But, improvements after putting such a system in place are not automatic, and it is the content rather than the platform which is the key. An advanced learning management system provides the tools – the learning objects and a mechanism for their presentation – for learner-centered courses. However, these tools must be coordinated in a skillful way to achieve an optimal result. If the instructor putting the pieces together does not know the meaning and purpose of all these objects, and how they are used together, then the end result is likely to be no better than an implementation using more conventional methods. It is the *design of the content*, not the design of the platform, which is most critical. This situation presents a tremendous opportunity for an innovative system design that concentrates on the needs of the curriculum creator.

4.6 Professional Development Communities

Several research groups have developed online communities for teacher professional development. For instance, the Inquiry Learning Forum (Barab, MaKinster, & Scheckler, 2003) is an online community of practice for science teachers attempting to integrate inquiry learning into their classrooms. They provide video lessons, discussion forums, and sample labs that teachers can use in class. Similarly, the Mentored and Online Development of Educational Leaders for Science group at UC Berkeley (Corliss, Spitulnik, Higgins, & Kirkpatrick, 2007) works to help teachers include the WISE environment in their classrooms. They take a combined approach, supporting and mentoring teachers face-to-face in classrooms and providing online community tools to allow group members to discuss issues they've faced with the integration process. A final example of this sort of tool is EdTech Leaders Online (Treacy, Kleiman, & Peterson, 2002), a community of educators working to incorporate technology into classrooms.

All of these communities are primarily targeted at primary- and secondary-school educators, rather than my goal of reaching tertiary-level instructors. It is apparent that professional development at the university level is a more difficult problem to solve, as faculty members are not primarily paid to teach or think about optimal curriculum design. Additionally, most extant professional development communities focus on very specific domains of interest and work primarily to foster a stand-alone community of teachers. By contrast, I am interested in developing rich community collaboration tools with a specific emphasis on sharing best practices in the form of pattern annotated courses. I believe that this focuses my design more directly on the problem of developing learner-centered curricula.

Chapter 5

PACT Generation One: Prototype Design, Applications, & Evaluation

The PACT project started with a simple goal: to expose experts' thoughts on learner-centered curriculum design and best practices within the context of real curriculum design artifacts. I began the process of achieving this goal in two ways: first, by launching an in-depth contextual inquiry process to study the way that real curriculum designers work and how they communicate their knowledge to other designers; second, by creating a speculative prototype curriculum design tool that incorporated a simple pattern annotation mechanism that allows users to label portions of a design with relevant pedagogical patterns. These two processes, the contextual inquiry study and the iterative refinement of the design tool, would eventually span approximately six years of work. The results of these processes informed my formulation of Virtual Design Apprenticeship (as described in Chapter 2) and led to the eventual creation of the final version of the PACT design tool (described in Chapter 7). In this chapter, I discuss my contextual design process, the initial design of the PACT tool, applications of the tool, and the lessons I learned in this stage of the project.

5.1 Contextual Design Process

I began my initial investigation of learning situated within design tools with a contextual inquiry process. I worked closely with the UC-WISE Computer Science Education research group in the EECS Department at UC Berkeley, whose aim is to improve the quality of computer science curricula and the instructional platforms used to deliver it. This team worked with university computer science instructors striving to adopt new technologies – and corresponding appropriate teaching practices for those technologies – in the classroom. Throughout my contextual design process, I participated in many dozens of meetings and curriculum design sessions which focused on iteratively improving both lab-centric courses and the UC-WISE system. Over the course of my six year involvement in these meetings, participants included about 10 experts on computer mediated learning and lab-centric instruction, approximately 10 experienced computer science instructors with no CML expertise, several education researchers, and more than 20 undergraduate and graduate student novices who were building their understanding both of computer science content and how to use Lab-Centric Instruction to convey it.

My goal in these sessions was to capture expert knowledge of LCI and the ways in which experts convey this knowledge to other participants. The understanding I built over the course of those six years informed the initial formulation and iterative improvement of the VDA learning model (discussed in Chapter 2), the initial design and implementation of the standalone PACT curriculum design tool (presented in the remainder of this chapter), and, eventually, the final design and implementation of the PACT overlay design tool (described in Chapter 7).

5.2 Initial Design

I initially designed PACT in a relatively naïve way – knowing that I wanted to expose expert thinking in the form of pedagogical pattern annotations, but little else. I began with Microsoft Visio and other visual layout tools as my primary inspiration, seeking to build visual representations of course contents with tools for editing and annotating them. PACT was intended to be engaging, fun to use, and valuable in the design process; ideally making it easier to make the right curriculum design than to make the wrong one. My design was 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. I had not yet formulated the Virtual Design Apprenticeship learning model while working on this iteration of PACT, but some of VDA's design principles are (at least partially) realized by features in the PACT interface – these connections are pointed out throughout this section. I refer to this initial iteration of the PACT design tool as PACT1 throughout this chapter, to distinguish it from the final iteration discussed as PACT2 in Chapter 7.

The primary perspective in PACT1 is course view, the features of which are described in the following sections. This display (Figure 5.1) shows the learning objects comprising a course connected to references to pedagogical patterns that indicate why design choices were made. These core artifacts of PACT are reifications of both course content and the instructional expertise that was needed to create it. The encapsulation of these two concepts is considerably more powerful than either one alone. These pattern-annotated courses are put into PACT by expert instructors who designed the course with solid pedagogy in mind and have taught the course over several iterations.

A pattern-annotated course is a very useful reference when creating new curriculum and is simple to modify to suit a new situation or instructor. Novice instructors learn about pedagogical patterns and their value by examining their uses in sensitizing examples – courses that the instructor can relate to in a meaningful way. The process of annotating a course with pattern references in PACT is also useful to experienced instructors as it encourages deep reflection and consideration of course goals and pedagogical methodology. Successive refinements over the course of several semesters can be undertaken in a structured manner using PACT.

5.2.1 Direct Manipulation Design Tool

The course view display in PACT1 is a collection of content objects and annotation objects (Figure 5.1). Each of these elements is directly manipulable – that is, they can be adjusted by interacting directly with them using a mouse. Users can easily create and delete objects, reorganize learning objects and pattern references, and build links between objects and patterns with these basic mouse operations. Connecting the behavior of course objects directly to the user's input further reifies what were once abstract concepts. The sleek simplicity of rearranging course elements helps to encourage experimentation with a variety of course/pattern configurations and provides an

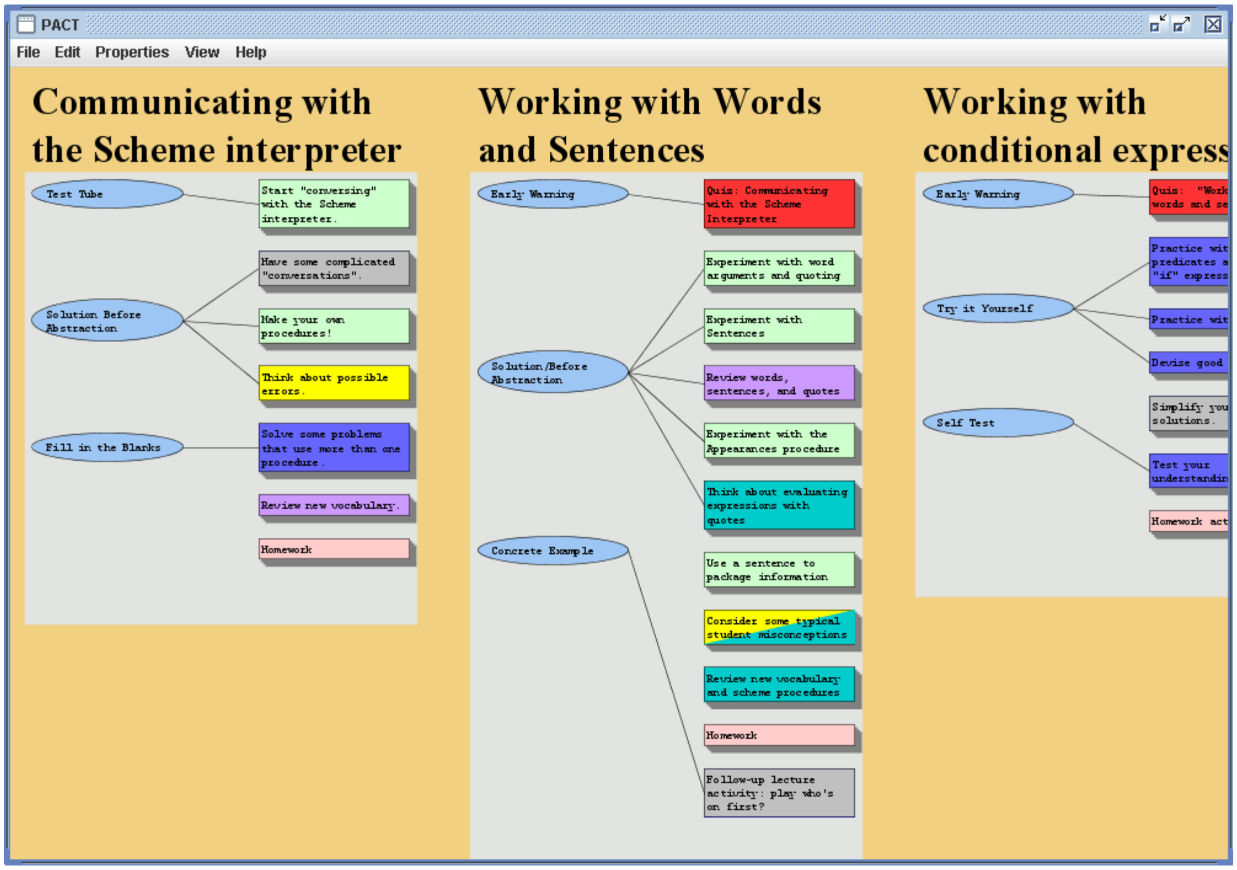

PACT
File Edit Properties View Help
Communicating with the Scheme interpreter
Test Tube
Start "conversing" with the Scheme interpreter.
Have some complicated "conversations".
Solution Before Abstraction
Make your own procedures!
Think about possible errors.
Fill in the Blanks
Solve some problems that use more than one procedure.
Review new vocabulary.
Homework
Working with Words and Sentences
Early Warning
Quiz: Communicating with the Scheme Interpreter
Experiment with word arguments and quoting
Experiment with Sentences
Solution/Before Abstraction
Review words, sentences, and quotes
Experiment with the Appearances procedure
Think about evaluating expressions with quotes
Use a sentence to package information
Concrete Example
Consider some typical student misconceptions
Review new vocabulary and scheme procedures
Homework
Follow-up lecture activity: play who's on first?
Working with conditional express
Early Warning
Quiz: "Work words and se
Practice wit predicates a "if" express
Try it Yourself
Practice wit
Devise good
Simplify you solutions.
Self Test
Test your understandin
Homework act

Figure 5.1: Three days of annotated content in PACT1's course view. Large grey boxes are days—each day contains all of the material a student will work through in a lab session. Smaller rectangles are activities, coherent sequences of learning objects grouped together under a common title. Activities are color coded based on the types of content they hold. Blue ovals are pattern annotations, with the text of the annotation indicating what pattern is being instantiated.

incentive to instructors who want to adapt existing courses to their own needs. These core features are what make a design tool a design tool, and serve as an instantiation of the first VDA principle to **enable design artifact creation and annotation.**

5.2.2 Intuitive Navigation

The course view display is a Zoomable User Interface (ZUI) (Bederson, Grosjean, & Meyer, 2004). The ZUI metaphor makes navigation between different scopes within a course (e.g. an activity, a day, an entire semester) simple. Users can easily "dive in" to the details of a short class segment or see the entire lay of the land from far above. This powerful visualization brings a curriculum out of the abstract and gives it a sense of physical structure. The association between this structure and pattern annotations helps bring the patterns into the concrete. Most importantly, the ZUI interface makes it easy for users to keep track of their location in the course as a whole while working on editing a small portion of the course, thanks to the context provided by surrounding objects. This focus plus context display technique (Baudisch, Good, Bellotti, & Schraedley, 2002) helps ease cognitive burden while navigating. This feature is a basic realization of the VDA principle to **make artifact exploration and comprehension easy**.

5.2.3 Informal Representations

Instructors must be willing to make changes to existing courses to get the full benefit of PACT. Unfortunately, many existing diagramming tools use "presentation-style" graphics such as 3D objects, intricate shading, and elaborate fonts. These visual elements suggest a highly evolved or final diagram. PACT uses diagram representations that feature solid boundaries, informal fonts, and playful colors. This style suggests an intermediate and abstract sketch and has been shown to encourage change and exploration (Hong & Landay, 2007).

This feature of the PACT1 design is a basic realization of the VDA design principle to **encourage design experimentation but preserve expert design intent**, effectively covering the "encourage design experimentation" portion but not the "preserve expert design intent." This issue would be addressed in the design of PACT2 (as described in Section 7.1).

5.3 PACT1 Implementation Details

PACT1 was written in Java and is primarily based upon the open source Piccolo ZUI library. The Course View and Pattern View interfaces are both generated using a static layout algorithm which uses fixed rules for where to place interface elements and how to size them. The default layout of course materials and annotations is the only layout available. PACT1 supports a fixed collection of curricular object types based on the step types available in UC-WISE, but the content typing mechanism is not otherwise extensible. The data persistence mechanism in PACT1 is a collection of XML files, each of which represents a course or collection of annotations. While PACT1 does not support directly interacting with learning management systems, I did develop an import tool which allowed users to bring the contents of a UC-WISE course into PACT for annotation, perusal, and presentation.

The full source of PACT1, excluding external libraries, is approximately 5,000 lines of code.

5.4 Applications of PACT

After completing the initial PACT1 implementation, I saw that it had immediate potential to help both inexperienced and experienced course designers throughout their design process. Additionally, I saw 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, I describe several usage scenarios in which PACT and pattern annotations can shed new light on the design process and expose the underlying educational theory behind good instructional designs.

5.4.1 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. My 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.

I 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 Section 5.5, 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, holistic picture. Intuitive navigation and editing combined with a highly visual information presentation mechanism 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.

5.4.2 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 cannot be achieved in the abstract. Through iterative cycles of working with expert-created examples and blending them into new designs, PACT can teach the novice instructor about learner-centered pedagogy using course contents that *she designed* – a highly motivating example. This process is examined in detail in Chapters 7 and 8.

5.4.3 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 designers (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 (Figure 5.2).

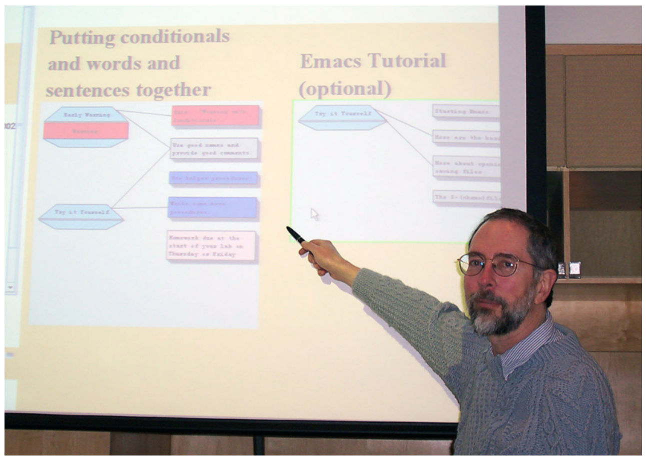

Putting conditionals and words and sentences together
Emacs Tutorial (optional)

Figure 5.2: UC-WISE Research Group leader Michael Clancy references the PACT display during a group meeting.

The UC-WISE research group used PACT in this capacity to structure the iterative improvement of CS61BL, a basic data structures course. They 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 learning activities being discussed within 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 comparatively little need to discuss *why* an activity exists, as the patterns help to describe this visually.

5.5 Sample Annotation Process

In this section, I will describe a model annotation process conducted by an experienced course designer and the lessons learned from the process. The course of interest is UC Berkeley's introductory Scheme-based programming course for non-computer science majors, CS3L. CS3L is delivered in the UC-WISE integrated learning environment (M. Clancy, Titterton, Ryan, Slotta, & Linn, 2003) which provides a variety of tools for student interaction.

These tools include:

- Pages of Web text, sometimes used for explanation, sometimes to present an 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 or discussion.
- A personal journal for information and reflection.

Step #	*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 'mazzanine) and (first '(mezzanine)), and between (first (square 7)) and (first '(square7)).
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?

Table 5.1: A partial day's worth of CS3L activities with their UC-WISE step types.

Table 5.1 lists the use of these tools in a portion of the second day of lab activities for CS3L. 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.

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

Table 5.2: Labeling steps from Table 5.1 with pattern references.

Michael Clancy, who designed the CS3L UC-WISE curriculum in 2002, undertook an example annotation of the CS3L 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.

These patterns are what I refer to as *ad hoc* pedagogical patterns – patterns that an instructor believes to be useful in describing their own design, but that have not been formally validated by pattern researchers (such as is done in the process used by the Pedagogical Patterns Project).

By systematically stepping through the curriculum, Clancy was able to identify a number of instances of each of these patterns. Table 5.2 lists the instances of these patterns found within the portion of the CS3L curriculum described in Table 5.1. Figure 5.3 shows a portion of this segment annotated in PACT. As it turned out, the vast majority of the curriculum in CS3L could be described using a combination of these two patterns. 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 pattern-annotated artifact.

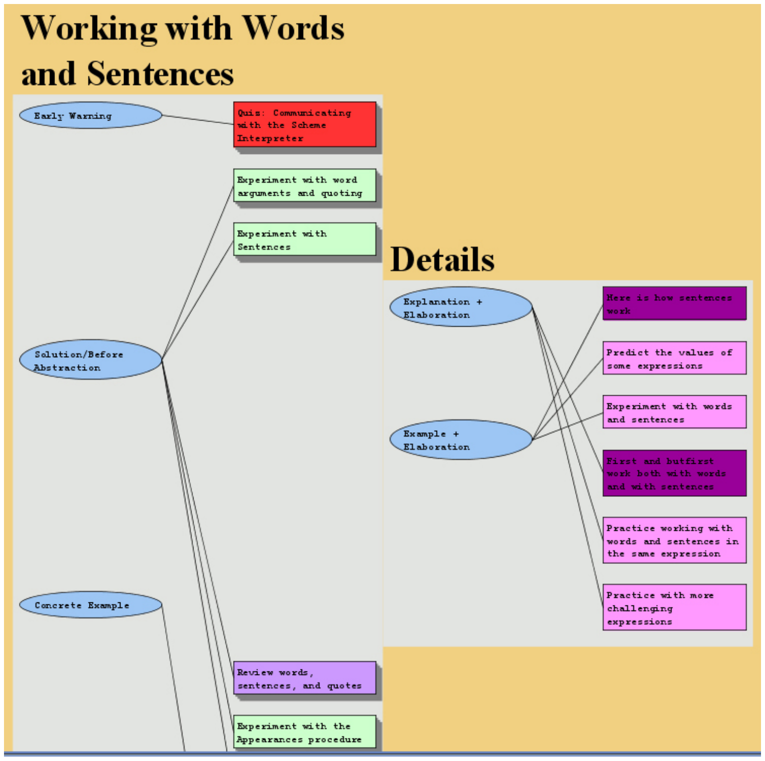

Working with Words and Sentences
Early Warning
Quiz: Communicating with the Scheme Interpreter
Experiment with word arguments and quoting
Experiment with Sentences
Solution/Before Abstraction
Concrete Example
Review words, sentences, and quotes
Experiment with the Appearances procedure
Details
Explanation + Elaboration
Example + Elaboration
Here is how sentences work
Predict the values of some expressions
Experiment with words and sentences
First and butfirst work both with words and with sentences
Practice working with words and sentences in the same expression
Practice with more challenging expressions

Figure 5.3: A close-up of one day of material in PACT1's course view. An activity has been opened to show the learning objects inside.

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 example code	Rewrite earlier
	Use the code
	Rewrite the code
	Explain how the code is similar to or differs from another program segment

Table 5.3: Common instances of the sample patterns.

5.5.1 Utility of the Annotation Process for Experts

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 5.1 is activities 18-19, where students are first exposed to the activity of analyzing a mystery function. This provokes a question to the reflective designer: 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 the UC-WISE team was cognizant of these repeating patterns, a second avenue of inquiry was apparent. Many instances of the two patterns took the form of one of the examples listed in Table 5.3. With the pattern and its pedagogical intent in mind, one can see that it is critically important that each of the "explanation" activities be easily understood by each student. 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. This led the group to ask themselves: 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 of certain activities in CS3L by the curriculum development group. This discussion would never have taken place if not for the annotation of CS3L and subsequent analysis. I, and the UC-WISE group, feel that similar analyses are critical to the development of robust learner-centered curricula for other courses, classroom platforms, and pedagogies.

5.6 Lessons from Contextual Studies

My formulation of the VDA framework and design of the PACT curriculum design tool were not solely guided by this formative study of the utility of PACT in the hands of experts. In addition to these findings, I also incorporated a number of field observations from my contextual design process with the UC-WISE group. In this section I summarize those findings, with an emphasis on the results that directly impacted VDA and the second iteration of PACT.

Experts and novices think about the curriculum design process, and evaluate the relative strength of professional designs in fundamentally different ways. Novices generally thought about a curriculum in terms of its individual activities. When asked to improve a curriculum, they selected individual components that they felt were weak and attempted to improve them in isolation. Experts, by contrast, thought about curricula as targeted sequences of activities. To an expert, a good curricular sequence carefully guides the learner through a series of questions, discoveries, difficulties, solutions, and opportunities for reflection. Crafting this overall structure is the essential element of creating good courses and has the largest effect on student learning—optimizing the individual components of the structure will have a much smaller impact. This observation informs an assumption of VDA: *focus on efficiently communicating the structural aspects of a design rather than the details of its components.*

Another key observation is the frantic pace of the curriculum development process. Experts and novices alike created new curricula or iterated over existing designs only shortly before they went in front of students. In one extreme example, a highly experienced instructor woke up at 4 a.m. each morning to build the content for his 10 a.m. lab section. This observation has two main implications for VDA and PACT. First, time constraints make it impractical to have separate learning tools and production tools—*learning must be integrated into the production tool itself.* Second, *tools must make it easy to quickly do the right thing.* It should be easier to produce a good design than a bad design.

Also, context is critical when experts and novices discuss, debate, and critique curriculum designs. One common practice in curriculum design sessions was to print out outlines of large sections of a course and distribute them to all participants. When inventing a new course, these printouts from existing courses served as models for new sequences. When iterating over existing designs, the outline showed surrounding material that connected to the section needing redesign. The group rarely discussed a change to individual activities or lessons in isolation. Significantly, the group also did not commonly discuss abstract best practices in isolation; discussions of recurring design principles or optimal ways to structure materials were nearly always couched in the context of a concrete design. In developing VDA, *preserving this context-rich mode of discussion is crucial.*

Finally, experts in learner-centered curriculum design took on a secondary role as publishing researchers in their field. For instance, computer science instructors published their new course designs at the ACM's SIG-CSE Symposium. When publishing their work, it is useful to present evidence from deployments of their new curricular idea, preferably in a range of contexts. Thus, the experts are motivated to distribute their course content to a number of other instructors, and then gather data about the content's effectiveness. This presents an ideal opportunity to match up experts and novices, offering incentive for experts to provide help to the novices, as long as the novices return the favor by providing data back to the expert. VDA seeks to *exploit existing incentive structures.*

Chapter 6

Formalizing Design Patterns as Annotations

My initial investigations using PACT1 demonstrated that searching through an existing curriculum and applying annotations to recurring patterns in an *ad hoc* manner, while possible, fails to produce annotated artifacts that fully deliver on the potential of pattern-annotated courses. Annotations like example + elaboration begin to tell the story of what the course designer was thinking, but they do not present a comprehensive and holistic picture of the strengths of a specific course design. These *ad hoc* annotations lack the formality, structure, and coverage of the curriculum design space that are provided by a robust pedagogical pattern language.

Further, the identification and application of these *ad hoc* patterns is a fundamentally personal process. Only the creator of the course can reliably indicate what she was thinking while engineering the curriculum – it is essentially impossible for non-creator annotators to build accurate and interesting pattern-annotated courses using *ad hoc* patterns. And while the patterns created intuitively by an expert curriculum designer are compelling (as discussed in Chapter 5), they lack rigorous theoretical and empirical evidence of their accuracy, pedagogical value, and general applicability. Building the evidence to validate a pedagogical insight as a full-fledged pedagogical pattern requires considerable research effort which is often beyond the interests and individual abilities of expert curriculum developers.

A final strike against *ad hoc* pattern annotations is that they are very difficult to reason about and to verify, both for human readers and computationally. When an annotator places a pattern on a piece of curriculum, the reader must either trust that the annotation is accurate or do considerable work to verify its accuracy – first developing a thorough understanding of the pattern being instantiated and then carefully examining the curriculum segment to ensure that it matches the claims of the pattern. This effort breaks the flow of learning, requiring the novice to jump in to highly abstract material before returning to the concrete work at hand.

I conclude that *ad hoc* pattern annotations are suitable for three tasks: (1) helping expert designers iterate over and rationalize about their own content; (2) identifying potential pedagogical patterns that merit further investigation; and (3) communicating design ideas internally within a curriculum design or research group. They are not, however, ultimately useful for communicating design principles to novice curriculum designers.

A more formal, structured approach to creating pattern-annotated courses is necessary to address these concerns. Based on my initial deployment of PACT and my contextual inquiry process, I determined that a reasonable set of design principles for such an approach are:

- The vocabulary of available annotations must be derived from a fixed repository of pedagogical patterns that have been empirically or theoretically validated and that purport to comprise a pattern language that covers the space of possible curriculum designs within a context. This requirement is to ensure that I am providing sound advice to novices that they can both trust and easily verify.
- The annotation schema must be easy to apply to content both for the expert curriculum designers that made the content and for trained annotators uninvolved in the content creation process. This requirement is necessary to allow pattern-annotated courses to be created in situations where the expert curriculum designer is not herself interested in contributing to the annotation process.
- The schema must define clear links between pedagogical pattern annotations and the properties of the instantiating content that qualify it as an instance. This requirement serves to make the relationship between patterns and content more salient and useful, letting annotations serve as scaffolding in the learning process for novice designers.

6.1 The Pact Annotation Schema & Library

To satisfy the design principles listed above, I have developed the PACT Annotation Schema (PAS), a formal mechanism for decomposing the free-form narrative of a pedagogical pattern description into annotations – concrete elements that can be identified and labeled in a course. I developed PAS through an iterative process of reflection and incremental formalization of an existing pattern repository: the 74 publicly available patterns created by the Pedagogical Patterns Project (Sharp, Manns, & Eckstein, 2000). The outcomes of this process were twofold, producing both 1) the PAS method for creating formal annotations based on free-form pattern descriptions and 2) a library of annotations that can be used in PACT.

The given format for each pattern in the Pedagogical Patterns Project repository is a semi-structured block of text, typically about one page in length. I began the process of converting these semi-structured descriptions into formal representations by writing concise (two- to three-sentence) summaries of each pattern. These summaries were used for my own reference and embedded into PACT to be displayed to users, as described in Chapter 7. Then, I carefully examined each pattern for descriptions or implicit references to specific types of content that the pattern prescribes to be presented to students. These content types became PAS Tags, as described below. Often, content types had to be inferred from vague descriptions of ideal student activities. Throughout this process, I worked diligently to ensure that my formalization of the content types did not interfere with the pedagogical intentions of the original pattern text.

Next, I examined the pattern text for details on how the pattern should be implemented. Some patterns describe a sequence of ordered steps to be undertaken within the curriculum. Others prescribe only that a specific type of content is desirable and should be noted as significant or employed as frequently as possible. Finally, some patterns describe specific activities or features of a course in highly specific detail that is difficult to generalize to other activities. Each of these pattern types is described in detail below. It is important to note that *these formalizations are not explicitly stated* *in each pattern* – they are derivations of the pattern that I have created to make them more useful within my design context and are a considerable contribution to the field of pedagogical design patterns.

6.1.1 Tags

The tag is the fundamental element within my annotation schema. Tags are applied directly to individual pieces of content and describe the content's pedagogical purpose or role within the curriculum design. Unlike tags in the general sense (e.g., free-form descriptive labels, such as those used to categorize photos on Flickr (Nov, Naaman, & Ye, 2008)) tags in my schema form a fixed, finite set that is derived from the patterns that comprise a pedagogical pattern language. The collection of tags corresponds directly to the conceptual design space covered by the pattern language.

As such, these tags are derived directly from the free-form text of a pattern description. For example, the Self-Test pattern (Bergin, Eckstein, & Sharp, 2002a) fundamentally recommends that the introduction of a new topic should be followed by an activity that provides feedback to students on how well they understand the concept (often taking the form of a quiz or exercise). Thus, the Self-Test pattern implies two tags that describe unique pedagogical roles that content can take on: Introduction and Feedback. By annotating learning activities that fulfill these roles, I can begin to understand why each piece of a course exists and, further, begin to discover portions of the course that might embody a Self-Test pattern. By repeating this process for each of the patterns of the Pedagogical Patterns Project, I derived a collection of 56 tags (Table 6.1). In aggregate, these tags describe the collective design space of potential learning activities envisioned by the authors of the Pedagogical Patterns Project's patterns.

Within PAS, tags are hierarchical. Many patterns specify only a fairly generic type of content (for instance, Feedback in the aforementioned Self-Test pattern). Many types of learning activity could provide the desired Feedback quality, for instance a Creation Exercise or Peer Feedback activity (itself often actually instantiated by more specific tags) could suffice. These relationships are noted in Table 6.1.

Content: This tag is implicitly applied to all content in a course, for use by patterns that are agnostic to content type/contents.
Active Content: Active Content is when a student is actively involved in constructing knowledge. Building, coding, experimenting. Not reading, watching videos, or listening to lectures.
Optional Content: Content that is not a required element of the course. May contain material intended for advanced students, but not essential for understanding.
Introduction: The introduction of a new concept or idea.
Review: Re-covering material that was already covered earlier in the course. This can be done to refresh the student's memory, to provide context for new material, or in preparation for exams.
Example: Examples show a practical application of a concept or technique. Examples can range in scale from small snippets of text or code up to large, complex artifacts.

Exercise: Practice or training that puts new concepts to work and allows the student to check their understanding and improve their ability to reason about and employ the new ideas.
Feedback: Any content that aides the student in assessing their own level of understanding of a concept. This could be as simple as revealing the solutions to an exam or as complicated as an online, distributed discussion forum in which consensus about the correct answers and approaches for a topic is collaboratively built. For feedback that is tailored to the individual student, tag with Differentiated Feedback.
Differentiated Feedback: Content that provides feedback tailored to the needs of each individual student. Examples include returning graded exams and assignments, one-on-one meetings with a TA, and auto-graded quizzes and assignments. Acts as: Feedback
Self Test: An opportunity for students to independently and autonomously assess their own level of understanding of material. Ideally, this level of understanding is captured in a way that can be easily presented back to the course staff, such that they can use it to gauge the overall understanding of the class. Acts as: Feedback
Take Questions: An opportunity for students to ask questions. Useful both for learning and for monitoring student progress. Acts as: Feedback
Peer Feedback: An explicit opportunities for students to give each other feedback on their work. Optimally, a firm structure is provided such that even introverted students can meaningfully provide feedback to their peers, and so that no one comes out of the experience feeling attacked or put down. Acts as: Feedback
Participant Feedback: An opportunity for the student to provide feedback to the course staff.
Easy Exercise: Exercises that all students should be able to complete successfully. Acts as: Exercise
Medium Exercise: Exercises that strong students can complete easily but that require more effort for struggling students. Acts as: Exercise
Hard Exercise: Exercises that will challenge even strong students. May be too difficult for struggling students to complete without assistance. Acts as: Exercise
Impossible Exercise: An exercise that is literally impossible for the students to complete in the time provided. Ideally, such problems should appear straightforward, even easy. But their impossibility should become clear with effort and further comprehension. Acts as: Exercise
Experiment: An opportunity for students to discover answers to interesting questions on their own by probing the properties of a system or creating tools to do so. Acts as: Exercise
Practice: An exercise that practices a recently taught skill or relates to a new concept. Good practices should help the student realize when their understanding is lacking. Acts as: Exercise
Creation Exercise: An exercise that requires creating something, rather than just testing comprehension. Examples include programming, writing specifications or documentation, proofs, explanations, etc. The scope of a creation exercise can range from a small programming task to creating large artifacts. Acts as: Exercise
Comprehension Exercise: While it is generally best to engage students with exercises that involve creation of artifacts, some patterns can make good use of comprehension exercises. These exercises require an understanding of a concept or artifact, and test aspects of that understanding. Ideally, these exercises will help students gauge their own level of understanding. Acts

as: Exercise
Correction Exercise: An exercise in which students must correct small flaws in an otherwise sound artifact. This could include correcting bugs in code, fixing small design flaws in a system, etc. Acts as: Exercise
Fill in the Blanks: An exercise in which students must add small parts to a large artifact in order to complete it or add additional functionality. Acts as: Exercise
Revision Exercise: An exercise that has the student revising an artifact that they previously created, usually in response to having received some form of feedback on their work. Acts as: Exercise
Critique Exercise: An exercise in which the student must understand and evaluate an artifact made by someone else (e.g. a fellow student, the course staff, professionals, etc.). The student should critique the artifact and explain its strengths and weaknesses. Acts as: Exercise
Multi-Topic Exercise: An exercise that touches upon more than one topic or concept, allowing students to test their understanding of more material without the overhead of understanding the setup for multiple exercises. Acts as: Exercise
Build a Tool: As an exercise, ask students to create a small, self-contained, reusable tool. Examples include small program portions that implement common algorithms or data structures, widgets/gadgets for computing useful formulas, etc. The focus on reusability should be motivated by an explicit intention to reuse the tool again later in this course or in subsequent courses. Acts as: Exercise
Toy Box: Provide students with programs/apps that emulate the functionality of more complicated real-world systems (e.g. a cache simulator). Have the students play with the program to get a feel for how the system works, without having to focus on its implementation details. Acts as: Exercise
Mutli-Topic Example: An example that is relevant to more than one topic or concept, allowing students to cover more material without the overhead of understanding many examples. Acts as: Example
Artifact Examination: Students can learn a lot about how experts design and build real-world artifacts by examining those artifacts (e.g. large programs, designs, documentation, etc.). Students need not be able to comprehend every detail of the artifact â€“ only to attempt to develop and understanding of how it works and why it was made the way it was. Acts as: Example
Lay of the Land: Give students a real-world example of the type of artifact that professionals in their field are expected to be able to create. Give it to the students to consider and to help them appreciate the scope of the field they are studying. Acts as: Example
Metaphor: An analogy or metaphor that maps a newly learned concept onto the conceptual space of something that the student already understands. Acts as: Example
Physical Analogy: A physical analog to an abstract concept. Used to engage students and make the abstract notion more concrete. Acts as: Example
Define a Continuum: Discuss the solution space for a problem relevant to the topic at hand. Identify the characteristics of that space and the extreme solutions at the edges of the space. Acts as: Example
Pair Exercise: Exercises in which a pair of students collaborate to devise a solution. The exercise

should be constructed such that peer learning can occur between the students.
Small Group Exercise: Exercises in which small groups of (3 â€“ 5) students work collaboratively to devise a solution. The exercise should be constructed such that peer learning can occur among the students.
Group Exercise: Exercises in which large groups of (> 5) students work collaboratively to devise a solution. These exercises must be carefully formulated to ensure that all students participate equally. Often, distinct roles must be assigned to individual students to manage the organization of the group.
Peer Learning: An opportunity for students to learn from the experiences and knowledge of one another. Could be a chance to share concern, questions, insights, ideas, etc. Peer learning can occur in dyads, small groups, large groups, or in a round-robin fashion among the class.
Round Robin: Soliciting discussion and input from as many of the students in the class as possible. In traditional classrooms, this often means calling on some random sample of the students present, and changing whom is called upon on each occasion. With online learning methods, it is possible to get every student to respond to every question and participate in every discussion.
Reflection: An explicit opportunity for students to pause and reflect upon what they have learned, what was difficult or easy, and why it was important. Reflection is key to metacognitive development and for helping students monitor their own learning.
Own Words: An opportunity for students to express a concept that they have just learned in their own words. Acts as: Exercise
Expand the Known World: Content that explicitly builds upon experiences that you know your students already had before the current course. (In cases where the material was covered within this course, Linking Old to New is the more appropriate annotation.)
Use a Tool: Allow students to use tools that they themselves built earlier in the course or in a previous course to solve a compelling problem.
Big Idea: A concept that is crucial to understanding the overarching themes and purpose of the course. Concepts that any student that successfully completes the course should absolutely be expected to understand thoroughly.
High-Level Overview: Material intended to provide a high-level overview of the topics in a course. Complex sub-topics should be avoided and no small details should be covered.
High Abstraction: Material that provides an in-depth overview of a topic or concept. The complexities of the topic should be exposed to students, but implementation-level details should be avoided.
Details: Material that covers specific implementation details of a topic or fine-grained engineering constraints and decisions. After completing this material, students should have a full understanding of a topic and how to create related artifacts.
Tiny Experience: Coverage of concepts at a topic-by-topic basis, with a high level of detail and relatively low level of abstraction.
Small Experience: Material that covers combinations of (tiny) topics into larger applications and settings.

Large Experience: Material that focuses on combining many small concepts together into one application that solves a large real-world problem.
Industry Partner: Classroom content that is presented in part by professionals engaged in the practice currently being studied. Guest speakers, presentation of real-world issues and exercises, etc.
Wider Perspective: Content that helps build an interdisciplinary understanding of the implications of the technology that students are learning to produce. Examples of such topics include policy, social norms, environmental concerns, economics, etc.
Learning Style: An activity that addresses a sensory modality or learning style that is not otherwise the prevailing learning style within the course. For instance, in a course dominated by reading and creating, a visualization or kinesthetic activity would qualify.
Visualization: Visualizations of complex systems or processes.
Real World Example: An example pulled from professional practice.
Reading Assignment: A reading assignment, to be done before the next class meeting.

Table 6.1: Tags in the PACT Annotation Library.

6.1.2 Multi-Step Patterns

Multi-step patterns are those that imply an ordered sequence of activities with specific tags, with each piece of content filling a specific role in the overall structure of the pattern. An example of a multi-step pattern is "Linking Old to New" (Bergin, Eckstein, Manns, & Wallingford, 2001), which can be summarized as, "Learning something new is exciting, but it often involves questioning things that the learner already knows. So, use an old wrapper to introduce new information, introducing new concepts in the context of material already learned." This implies two steps that form a pattern: Review and Introduction. In the Review step of the pattern, previously covered material that relates in some way to a new concept to be introduced is revisited and refreshed for the student. In the Introduction step of the pattern, the new material is presented, grounded within the context of the earlier content. Thus, Review and Introduction emerge as tags and Linking Old to New is a multi-step pattern. Table 6.2 summarizes the 22 multi-step patterns of the PACT Annotation Library.

Test Tube (Bergin, Eckstein, & Sharp, 2002b)	
Discovering an answer for oneself is more powerful than being told an answer. Therefore, after introducing an interesting problem, allow the students to answer questions about the concept via experimentation rather than just reading an answer.	
Content Slot: Introduction	Accepts: Introduction
Content Slot: Experiment	Accepts: Experiment
Required Distance Among Steps:	immediately adjacent
Try It Yourself (Bergin, Eckstein, & Sharp, 2002b)	
Putting new concepts to work in practice is an excellent way to ensure that they are really learned, not just heard/read. Therefore, after introducing a new concept, ask the students to perform an exercise that requires them to understand it well.	
Content Slot: Introduction	Accepts: Introduction
Content Slot: Exercise	Accepts: Exercise
Required Distance Among Steps:	immediately adjacent

Student Design Sprint (Bergin, Eckstein, & Sharp, 2002b)
Give students a problem, have them develop a solution in groups over 20 minutes, and then spend a few minutes to publicly (but anonymously) review the solutions. Critique these solutions. Combine pairs of teams into a larger team, modify the problem slightly, and have the new teams solve the new problem based on the pair of solutions they now have to the original problem.

Content Slot: Creation Exercise	Accepts: Creation Exercise
Content Slot: Small Group Exercise	Accepts: Small Group Exercise
Content Slot: Critique Exercise	Accepts: Critique Exercise
Content Slot: Large Group Exercise	Accepts: Group Exercise
Content Slot: Critique Exercise	Accepts: Critique Exercise
Required Distance Among Steps:	immediately adjacent

Adopt an Artifact (Bergin, Eckstein, & Sharp, 2002b)
A useful pedagogical technique is to have students build understanding of a concept, create an artifact that uses that concept, and then hand the artifact off to another student to extend and improve. Students will benefit from learning about how their peers approached the task and what they created.

Content Slot: Creation Exercise	Accepts: Creation Exercise
Content Slot: Critique Exercise	Accepts: Critique Exercise
Required Distance Among Steps:	near each other

Larger Than Life (Bergin, Eckstein, & Sharp, 2002b)
You want to expose learners to complex ideas and get them used to processing and comprehending large programs or designs. Provide students with a large artifact too large for students to fully comprehend its details in the time allotted. Then engage the students with exercises that rely on a large-scale conceptual or structural model of the artifact, rather than knowledge of its details.

Content Slot: Artifact Examination	Accepts: Artifact Examination
Content Slot: Comprehension Exercise	Accepts: Comprehension Exercise
Required Distance Among Steps:	immediately adjacent

Round and Deep (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003)
Allow students to work on problems that can be solved in a number of different ways from different perspectives. Then have the students share their understanding and perspective with the rest of the group.

Content Slot: Expand the Known World	Accepts: Expand the Known World
Content Slot: Round Robin	Accepts: Round Robin
Required Distance Among Steps:	immediately adjacent

Fixer Upper (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003)
You want your students to be able to manipulate artifacts that are larger or more complicated than they could realistically produce on their own. Give your students a fairly large artifact that is generally sound, but with carefully introduced flaws. Ask the students to repair and discuss the flaws.

Content Slot: Artifact Examination	Accepts: Artifact Examination
Content Slot: Correction Exercise	Accepts: Correction Exercise
Content Slot: Round Robin	Accepts: Round Robin
Required Distance Among Steps:	immediately adjacent

Tool Box (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003)
Have students build small, reusable tools as part of exercise solutions. Then encourage the student to use that tool again later in the course or in subsequent courses.

Content Slot: Build a Tool	Accepts: Build a Tool
Content Slot: Use a Tool	Accepts: Use a Tool
Required Distance Among Steps:	far apart

Own Words (Bergin, Eckstein, & Sharp, 2002a)
Students may be able to repeat material you have presented to them verbatim, but they may not have fully understood them. Therefore, invite your students to express a key idea they have just learned in their own words. It will then be more clear to you and the student whether they have really understood the concept.

Content Slot: Key Idea	Accepts: Content
Content Slot: Own Words	Accepts: Own Words
Required Distance Among Steps:	near each other

Self Test (Bergin, Eckstein, & Sharp, 2002a)
If your students don't understand what you have presented, they have a poor basis for moving forward. If you don't understand where your students are at, you can't adapt your course to meet their needs. Therefore, let the student answer a self-test on a new concept before moving on to the next concept.

Content Slot: New Concept	Accepts: Content
Content Slot: Self Test	Accepts: Self Test
Required Distance Among Steps:	immediately adjacent

Embrace Correction (Bergin, Eckstein, & Sharp, 2002a)
Students can learn a great deal by revising artifacts on which they have already received feedback and/or a grade. Give the students opportunities to improve their work.

Content Slot: Creation Exercise	Accepts: Creation Exercise
Content Slot: Feedback	Accepts: Feedback
Content Slot: Revision Exercise	Accepts: Revision Exercise
Content Slot: Feedback	Accepts: Feedback

Reflection (Bergin, Eckstein, Manns, & Wallingford, 2001)
Redirect students to use their own intellect. Do not let them be too dependent on passive information absorption. Give them opportunities to monitor their learning.

Content Slot: Content	Accepts: Content
Content Slot: Reflection	Accepts: Reflection
Required Distance Among Steps:	near each other

Spiral (Bergin, Eckstein, Manns, & Wallingford, 2001)
Organize the course to introduce all the high level topics to students without covering them completely at first viewing, then on each cycle of a spiral cover old topics in more depth and include additional topics and details.

Content Slot: High-Level Overview	Accepts: High-Level Overview
Content Slot: High Abstraction	Accepts: High Abstraction
Content Slot: Details	Accepts: Details
Required Distance Among Steps:	far apart

Linking Old To New (Bergin, Eckstein, Manns, & Wallingford, 2001)
Learning something new is exciting, but it often involves questioning things that the learner already knows. So, use an old wrapper to introduce new information, introducing new concepts in the context of material already learned.

Content Slot: Review	Accepts: Review
Content Slot: Introduction	Accepts: Introduction
Required Distance Among Steps:	immediately adjacent

Abstraction Gravity (Bergin et al., 2003) Introduce a concept at its highest level of abstraction and use reflection on the concept to link the higher-level abstraction to the lower one.	
Content Slot: High Abstraction	Accepts: High Abstraction
Content Slot: Details	Accepts: Details
Solution Before Abstraction (Bergin et al., 2003) Give the students an example of the problem in a setting that they are comfortable with before moving on to showing how aspects of the solution can be applied to similar problems. This approach can be used to provide motivation for learning a topic.	
Content Slot: Example	Accepts: Example
Content Slot: High Abstraction	Accepts: High Abstraction
Required Distance Among Steps:	near each other
One Concept, Several Implementations (Bergin et al., 2003) Use several different implementations of the concept as examples while teaching an abstract concept.	
Content Slot: Introduction	Accepts: Introduction
Content Slot: Example	Accepts: Example
Required Distance Among Steps:	near each other
Experiencing in the Tiny, Small and Large (Bergin et al., 2003) Introduce a complex concept in three stages, tiny, small and large, which allow you to monitor the student's progress on a topic-by-topic (tiny) basis, to test if the student can combine the topics and apply them in a larger (small) setting, and to solve a real-world (large) problem using all parts of the concept.	
Content Slot: Tiny	Accepts: Tiny Experience
Content Slot: Small	Accepts: Small Experience
Content Slot: Large	Accepts: Large Experience
Required Distance Among Steps:	near each other
See Before Hear (Bergin et al., 2003) Introduce students to a concept with hands-on experiences before explaining the concept in detail.	
Content Slot: Exercise	Accepts: Exercise
Content Slot: Introduction	Accepts: Introduction
Required Distance Among Steps:	near each other
Three Bears (Bergin et al., 2003) Ask the learner to create solutions that lie at both extremes of a continuum of possible solutions, as well as at some balance point. Then conduct a review that gives the learner an opportunity to reflect on the experience.	
Content Slot: Define a Continuum	Accepts: Define a Continuum
Content Slot: Creation Exercise	Accepts: Creation Exercise
Content Slot: Reflection	Accepts: Reflection
Required Distance Among Steps:	immediately adjacent
Mission Impossible (Bergin et al., 2003) Present the learner with a problem that seems straightforward, but whose complete solution requires a much deeper understanding than the basic concepts afford. In fact, it should be impossible in the time provided. After the exercise, dedicate time to reflecting on the problem and why it was more difficult than it appeared.	

Content Slot: Impossible Exercise	Accepts: Impossible Exercise
Content Slot: Reflection	Accepts: Reflection
Required Distance Among Steps:	immediately adjacent
Fill in the Blanks (Bergin, 2000)	
Ask students to create several small parts that fit into a given large artifact. Doing so will aid both their reading and writing skills.	
Content Slot: Artifact Examination	Accepts: Artifact Examination
Content Slot: Fill in the Blanks	Accepts: Fill in the Blanks
Required Distance Among Steps:	immediately adjacent

Table 6.2: Multi-step patterns in the PACT Annotation Library.

6.1.3 *Single-Step Patterns*

Single-step patterns are those that can be instantiated in a single learning activity. An example is Critique (Bergin, Eckstein, & Sharp, 2002b), summarized in PACT as: "Students should be able to understand and evaluate complex artifacts in addition to creating artifacts on their own. To practice this skill, give students a model or artifact created by someone else and ask them to critique it." This pattern prescribes an exercise (tagged in the PACT Annotation Library as Critique Exercise) which can, and often is, instantiated as a single learning activity. The success of this activity does not necessarily depend upon its surrounding context – the pattern is essentially agnostic to what happens immediately before and after the Critique Exercise. This is not to say that the Critique pattern *must* be instantiated in a single activity – just that it can be without violating the underlying pedagogical principle. Notably, there will always be a one-to-one correspondence between single-step patterns and the tags that describe their instantiations. Table 6.3 summarizes the 11 single-step pattern annotations in the PACT Annotation Library.

Students Decide (Bergin, Eckstein, & Sharp, 2002b)	
Engage the students in determining the contents and methods of the course. Provide optional material for more advanced students: "extra for experts."	
Content Slot: Optional Content	Accepts: Optional Content
Expand the Known World (Bergin, Eckstein, & Sharp, 2002b)	
A student's learning will be deeper if they associate a new concept to their existing knowledge and experience. Therefore, introduce a concept by linking it explicitly to experiences that you know the students already have. (In cases where the material was covered earlier in this course, Linking Old to New is the more appropriate annotation.)	
Content Slot: Expand the Known World	Accepts: Expand the Known World
Critique (Bergin, Eckstein, & Sharp, 2002b)	
Students should be able to understand and evaluate complex artifacts in addition to creating artifacts on their own. To practice this skill, give students a model or artifact created by someone else and ask them to critique it.	
Content Slot: Critique Exercise	Accepts: Critique Exercise
Wider Perspective (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003)	
The work of professionals is situated in a complex environment of economic conditions, organization policies and politics, social norms, environmental concerns, etc. Give students some sense of this complexity by building an interdisciplinary understanding of the implications of the	

technology they are learning to produce. Content Slot: Wider Perspective Accepts: Wider Perspective
Industry Partner (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003) It is desirable to give students an appreciation of how the topic they are studying is relevant and useful in the workplace, but fulltime academics do not always have a robust understanding of how workplaces function. Therefore, partner with one or more individuals from industry and include them in classroom activities. Content Slot: Industry Partner Accepts: Industry Partner
Peer Feedback (Bergin, Eckstein, & Sharp, 2002a) Students are able to give helpful feedback to their peers and both the giver and receiver of such feedback can learn from the experience. Unfortunately, students often lack the confidence to give useful advice in an unstructured manner. Therefore, give students structured opportunities to provide feedback on the work of their peers. Content Slot: Peer Feedback Accepts: Peer Feedback
Acquire Participants' Feedback (Bergin, Eckstein, & Sharp, 2002a) You have a one-sided view of your teaching style and can never be sure how well your style is received by students and how well this supports their requirements of a good learning environment. Therefore, invite students to provide feedback on your teaching style. Content Slot: Acquire Participants' Feedback Accepts: Participant Feedback
Consistent Metaphor (Bergin, Eckstein, Manns, & Wallingford, 2001) When introducing a new topic, consider using a large-scale analogy or metaphor to map the concept onto something the student already understands. Be consistent when doing so! Content Slot: Consistent Metaphor Accepts: Metaphor
Physical Analogy (Bergin, Eckstein, Manns, & Wallingford, 2001) It is relatively easy for learners to comprehend concrete concepts because they are usually easy to visualize. For more abstract concepts, create a physical analogue and engage your students using it. Content Slot: Physical Analogy Accepts: Physical Analogy
Toy Box (Bergin, 2000) Create programmatic elements that are analogous to more complicated (and detailed) real-world systems, then have students play with them in ways that allow them to understand how the systems work without focusing too much on the details of implementation. Content Slot: Toy Box Accepts: Toy Box
Lay of the Land (Bergin, 2000) Provide students a large artifact that they can study, but could not realistic come close to producing on their own. Examining this artifact should give students an appreciation for the field they are about to begin studying, without getting into implementation details. Content Slot: Lay of the Land Accepts: Lay of the Land

Table 6.3: Single-step patterns in the PACT Annotation Library

6.1.4 Trend Patterns

Trend patterns are those that state that specific types of content or activities should be done often or at a specific rate. A pervasive example of a trend pattern is Active Student (Bergin, Eckstein, & Sharp, 2002b), summarized in PACT as: "Students should be actively involved in constructing knowledge. Have as much active content as possible, and little passive content. Active content includes building things, trying things, experimenting, etc. Passive content is reading, watching videos, sitting in lectures, etc." Reading between the lines, we see that the concrete suggestion of this pattern is that *active learning should be done frequently* – ideally more frequently than passive content. This pattern gives rise to the Active Content tag (placed on all learning activities in which students actively construct knowledge, rather than passively receiving it). Thus, the pattern annotation becomes "Active Content appears on content in this day (or whatever is being annotated with the pattern) *frequently*." Table 6.4 summarizes the 13 trend pattern annotations in the PACT Annotation Library.

Active Student (Bergin, Eckstein, & Sharp, 2002b)	
Students should be actively involved in constructing knowledge. Have as much active content as possible, and little passive content. Active content includes building things, trying things, experimenting, etc. Passive content is reading, watching videos, sitting in lectures, etc.	
Content Slot: Active Content	Accepts: Active Content
Trend Base:	Content
Different Exercise Levels (Bergin, Eckstein, & Sharp, 2002b)	
Keep students of varied skill levels engaged and making progress by presenting exercises of varying levels of difficulty.	
Content Slot: Easy Exercise	Accepts: Easy Exercise
Content Slot: Medium Exercise	Accepts: Medium Exercise
Content Slot: Hard Exercise	Accepts: Hard Exercise
Trend Base:	Easy Exercise + Medium Exercise + Hard Exercise
Honor Questions (Bergin, Eckstein, & Sharp, 2002b)	
Encourage questions by taking every question seriously. Show participants how to ask good questions. Consider strategies to get uncomfortable students to ask questions, such as taking questions anonymously.	
Content Slot: Honor Questions	Accepts: Take Questions
Trend Base:	Content
Prefer Writing (Bergin, Eckstein, & Sharp, 2002b)	
Students learn best when creating new things, not just passively consuming information. Therefore, choose exercises that involve creation over those that only require comprehension. This could include writing programs, specifications, documentation, proofs, explanations, etc.	
Content Slot: Creation Exercise	Accepts: Creation Exercise
Trend Base:	Creation Exercise + Comprehension Exercise
Groups Work (Bergin, Eckstein, & Sharp, 2002b)	
Students need frequent feedback, but providing it directly or preparing auto-graded exercises can be time consuming. To lighten the load, emphasize group work and allow students to learn from each other.	
Content Slot: Group Work	Accepts: Pair Exercise, Small Group Exercise, Group Exercise, Round Robin
Trend Base:	Exercise
Invisible Teacher (Bergin, Eckstein, & Sharp, 2002b)	
The course staff is a finite resource that cannot always realistically address the needs of all students at all times. In general, students should be directed to ask their peers for help when possible. As a concrete learning activity, students could be given an explicit opportunity to share their concerns and questions with other students.	
Content Slot: Peer Learning	Accepts: Peer Learning
Trend Base:	Content
Multi-Pronged Attack (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003)	

Time in the classroom is scarce! Choose your examples and exercise so that they cover more than one idea or topic at the same time. Content Slot: Multi-Topic Content Accepts: Multi-Topic Exercise, Multi-Topic Example Trend Base: Exercise + Example
Early Bird (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003) Organize the course such that the most important topics are taught first. Teach the big ideas early and revisit them often. Note: This annotation should only be used with content early in the course! Content Slot: Big Idea Accepts: Big Idea Trend Base: Introduction
Feedback (Bergin, Eckstein, & Sharp, 2002a) To improve, students need tools to help monitor their progress. Work must be assessed and feedback given so that students can identify misunderstandings and incomplete knowledge. Therefore, give students feedback on their performance early and often. The best tools for this are exercise, tasks, and activities that challenge the student's understanding of new concepts and methods. Content Slot: Feedback Accepts: Feedback Trend Base: Content
Differentiated Feedback (Bergin, Eckstein, & Sharp, 2002a) Generic feedback is not as useful as feedback tailored to the needs of each individual student. Give differentiated feedback whenever possible. This can often be done automatically with auto-graded exercises. Content Slot: Differentiated Feedback Accepts: Differentiated Feedback Trend Base: Feedback
Early Warning (Bergin, Eckstein, & Sharp, 2002a) Help students establish whether or not they understood key concepts shortly after they learn them, especially if the concept will be essential for understanding material later in the course. Include plenty of self-tests, exercises, and other opportunities for the students to discover what they do and do not understand. Content Slot: Self Test Accepts: Self Test Trend Base: Content
Different Approaches (Bergin, Eckstein, Manns, & Wallingford, 2001) Not all students learn in the same way. Therefore, provide different approaches to the same topic that accept different learning styles by addressing various sensory modalities. Content Slot: Learning Style Accepts: Learning Style Trend Base: Content
Round Robin (Bergin et al., 2003) Class discussions lose their pedagogical value if they are dominated by a few individuals. Therefore, use a round robin technique to solicit suggestions from everyone in the class. Content Slot: Round Robin Accepts: Round Robin Trend Base: Content

Table 6.4: Trend patterns in the PACT Annotation Library.

6.1.5 *Special Patterns & Advice Patterns*

Special patterns are complex, abstract pedagogical techniques that could be realized in many different ways and with many different types of content. Formally, they are the set of pedagogical patterns from which tags cannot be derived. Often these patterns do not map cleanly onto the affordances provided within typical computer-mediated learning environments (such as those present in lab-centric instruction) or operate at a scope that is beyond the considerations of designing day-to-day learning activities. One representative example is the Real World Experience pattern (Bergin, Eckstein, & Sharp, 2002b), summarized in PACT as: "Some types of experience are impossible to replicate within the confines of the classroom. To have truly authentic experiences, students need to go to real workplaces and see what real work processes look like. Have the students work with outside experts on real world problems within real world situations." While instantiating this pattern would be a noble endeavor, it is difficult to see the utility of this pattern in the creation of lab materials and there is no reasonable mapping between lab learning activities and a realization of this pattern. Table 6.5 summarizes the 10 special patterns in the PACT Annotation Library.

Role Play (Bergin, Eckstein, & Sharp, 2002b) Complex systems and concepts can be difficult to understand, especially because they are often presented very abstractly in readings and lecture. One way of combating this is to distribute the details of the concept or system among a group of students and then have the students role play as a specific portion of the complete system. Content Slot: Role Play	Accepts: Content
War Game (Bergin, Eckstein, & Sharp, 2002b) It is difficult to tackle real-world problems and processes within the context of a classroom. A War Game is one way around this problem. Create time-compressed simulation games designed to highlight different roles and events in software development projects, distributing real-world roles among the participants. Content Slot: War Game	Accepts: Content
Shotgun Seminar (Bergin, Eckstein, & Sharp, 2002b) To enable a good discussion around a topic, the full audience needs to be well informed. If you want a student to give a presentation about the topic, but want to ensure that everyone understands it, try the Shot Gun Seminar. Assign all students in a group a research topic and then randomly select the student to present at presentation time. Keep the initial presentation short to allow time for discussion. Content Slot: Shotgun Seminar	Accepts: Content
Real World Experience (Bergin, Eckstein, & Sharp, 2002b) Some types of experience are impossible to replicate within the confines of the classroom. To have truly authentic experiences, students need to go to real workplaces and see what real work processes look like. Have the students work with outside experts on real world problems within real world situations. Content Slot: Real World Experience	Accepts: Content
Mock Exam (Bergin, Eckstein, & Sharp, 2002a) You want to help students prepare for an exam in a realistic context. Give the students a chance to prepare for the exam by permitting them to take a trial exam. Content Slot: Mock Exam	Accepts: Content
Explore for Yourself (Bergin, Eckstein, Manns, & Wallingford, 2001) Assign topics to students that they have to learn on their own then have them present the topic afterwards. Content Slot: Explore for Yourself	Accepts: Content

Build and Maintain Confidence (Bergin et al., 2003) Allow students to work out solutions to complex problems, providing hints via questions that have to be answered and that may lead to a solution. Content Slot: Build and Maintain Confidence Accepts: Content
Built-In Failure (Bergin et al., 2003) Remove the fear of failure as a barrier to learning by making failure an accepted, expected, and desired part of the learning process. Content Slot: Built-In Failure Accepts: Content
Expose the Process (Bergin et al., 2003) When showing examples or idealized solutions to exercises, also show and explain the process of getting there. Show the critical decision points to the students and allow them to make their own proposals on how to go on. Content Slot: Expose the Process Accepts: Content
Mistake (Bergin, 2000) Ask students to create an artifact (such as a program or design) that contains a specific error. Ask them to examine the consequences of this error and then develop a correction. Content Slot: Mistake Accepts: Content

Table 6.5: Special patterns in the PACT Annotation Library.

Closely related to the special patterns are advice patterns, which give general tips for managing a course or classroom. These patterns are not related to course contents and thus cannot be mapped onto learning activities and, like special patterns, do not give rise to any specific content tags. An example of an advice pattern is Team Teaching (Bergin, Eckstein, Manns, & Wallingford, 2001), summarized in PACT as: "Team up with fellow educators and teach the course together. Preferably, team up with someone that can provide a very different perspective from your own." It is clear that this pattern, while pedagogically sound, does not describe specific learning content and is of no assistance to an instructor trying to generate such content. Table 6.6 summarizes the 18 advice patterns in the PACT Annotation Library.

Study Groups (Bergin, Eckstein, & Sharp, 2002b) Form your students into study groups with complementing competencies and ability levels.
Teacher Selects Teams (Bergin, Eckstein, & Sharp, 2002b) Students tend to self-select into teams with their friends or people they are already comfortable working with. Unfortunately, this is not representative of the situations that professionals face in real jobs. Therefore, do not let your students self-select into teams -- select their teams for them in a way that will create some useful tension and growth.
Team Teaching (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003) Team up with fellow educators and teach the course together. Preferably, team up with someone that can provide a very different perspective from your own.
Nobody is Perfect (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003) Admit your limitations with grace. Do not try to be perfect. If you cannot answer a question, admit it.
Restructure (Bergin, Eckstein, Manns, Sharp, & Sipos, 2003) When concepts move from being fun extras to being more essential, promote them earlier in the course, effectively making them the new Big Ideas.
Challenge Understanding (Bergin, Eckstein, & Sharp, 2002a)

Give the participants exercises, tasks or activities that challenge their understanding of concepts.
Feedback Sandwich (Bergin, Eckstein, & Sharp, 2002a) When giving feedback, start and end with positive feedback, sandwiching suggestions for improvement between these reinforcing comments.
Student Online Portfolios (Bergin, Eckstein, & Sharp, 2002a) Students can get excellent feedback from their peers and others if their work is easy to obtain. Therefore, provide a means for students to publish their best work. The more public, the better.
Gold Star (Bergin, Eckstein, & Sharp, 2002a) By providing feedback to students exclusively in private, you lose the opportunity to show other students what you value most highly in student work. Therefore, when a student is doing well, or has done something well, praise them publicly for it.
Kinds of Exam (Bergin, Eckstein, & Sharp, 2002a) Use different kinds of exams which serve various student learning capabilities.
Fair Grading (Bergin, Eckstein, & Sharp, 2002a) You want to be fair in your grading and for students to be satisfied that they are being evaluated fairly. Therefore, publish your minimum grading standard and stick to it.
Key Ideas Dominate Grading (Bergin, Eckstein, & Sharp, 2002a) The key ideas, not necessarily the hardest material, should be worth the most points in your grading.
Grade it Again Sam (Bergin, Eckstein, & Sharp, 2002a) Permit your students to change and re-submit an assignment for evaluation and re-grading, after you have graded it and provided feedback.
One Grade for All (Bergin, Eckstein, & Sharp, 2002a) On small projects, all members of a group should be accountable for the end product. Give all members of a project team the same grade. Base that grade on a presentation that will be made by a random team member.
Fair Team Grading (Bergin, Eckstein, & Sharp, 2002a) On large projects, the complexity of working in groups can get in the way of giving all group members the same grade. Therefore, base part of project team members' grade on the team product, but part of it on individual contributions.
Peer Grading (Bergin, Eckstein, & Sharp, 2002a) You want to teach your students how to evaluate quality and how to negotiate for it. You want to get them to accept evaluation by peers and to make this comfortable. Therefore, make it possible for students to provide part of the grade for other students.
Fair Project Grading (Bergin, Eckstein, & Sharp, 2002a) Divide up the evaluation of large projects into different components, each of which will be given an independent grade.
Anonymous Feedback (Bergin, Eckstein, & Sharp, 2002a) Provide an anonymous feedback channel through which your students can communicate with you and encourage them to use it.

Table 6.6: Advice Patterns in the PACT Annotation Library

6.2 Pattern Constraints

Many patterns prescribe implementation details beyond specifying just what content types (tags) should be used in an instantiation. As already discussed, trend patterns specify frequencies at which content types should appear. This can be more complicated that simply specifying that content should happen "frequently" or "as often as possible" (as is the case with simple patterns like Active Student). Take, for instance, the Different Exercise Levels pattern (Bergin, Eckstein, & Sharp, 2002b), summarized in PACT as: "Keep students of varied skill levels engaged and making progress by presenting exercises of varying levels of difficulty." In the PACT Annotation Library, I have interpreted this pattern to imply three different tags: Easy Exercise, Medium Exercise, and Hard Exercise. The pattern is, thus, to have a *good mix* of these tags within the scope of content being annotated (usually a day or week worth of material). Formally, I count the total number of instances of each of those three tags and then specify a *portion* of the whole that should be Easy, Medium, and Hard, respectively. For each instance of this annotation, the specific ratios that make up the "good mix" for that particular context/portion of the course is left up to the annotator – PAS merely provides the opportunity for the annotator to specify it and make it a constraint upon the annotation.

Multi-step patterns often imply specific constraints with respect to the relative timing of each step in the pattern. For instance, the Reflection pattern (Bergin, Eckstein, Manns, & Wallingford, 2001) is summarized in PACT as: "Redirect students to use their own intellect. Do not let them be too dependent on passive information absorption. Give them opportunities to monitor their learning." This pattern specifies two tags (Content and Reflection) that should happen temporally close to one another. In contrast, the Tool Box pattern (Bergin, Eckstein, Manns, & Wallingford, 2001) – summarized in PACT as "Have students build small, reusable tools as part of exercise solutions. Then encourage the student to use those tools again later in the course or in subsequent courses." – specifies two tags (Build a Tool and Use a Tool) that should happen temporally distantly to one another.

6.3 Creating a Pattern-Annotated Course

I have developed three annotated course segments using the annotations contained in the PACT Annotation Library. Each course segment is a two-day (one week) piece of lab-centric content developed by the UC-WISE curriculum research group (primarily by Senior Lecturer Michael Clancy, Research Specialist Nathaniel Titterton, and Colleen Lewis, a PhD Student). The purpose of this exercise was three-fold. First, to demonstrate the utility of the annotation library and show that it can be used to create pattern annotated courses. Second, to show that someone with sufficient knowledge of pedagogy and pedagogical patterns, but not personally involved in the creation of the course materials, can successfully annotate a course. Third, to use as carefully constructed example annotated courses in the user study described in Chapter 8.

The process of annotating an existing course segment using the PACT Annotation Library is simple, if rather time consuming. I began each annotation by carefully reading over the entire course segment, noting high-level structures and getting a feel for the general flow of the materials. Next, I carefully examined each learning activity in the segment, attempting to determine the fundamental purpose or pedagogical role that each activity held in the overall sequence of activities. The goal of this stage is to map tags on to each piece of content. This is best done in an iterative manner, looking first for obvious tags (such as Active Content, Creation Exercise, or Optional Content) across all content and then returning to each activity to look for more subtle nuances (like High Abstraction, Low Abstraction, and Details).

Once the course segment was fully tagged, I began looking for patterns that can be annotated. The easiest to spot are the single-step patterns, as they follow directly from their content tags. Since single-step patterns and their tags share one-to-one relationships, it is up to the annotator to decide whether to emphasize a specific pattern by calling it out with its own annotation or if it is best left as just a tag. For instance, if a Critique Exercise occurs as one of many types of exercise mixed in to a day of material, it may not be worth highlighting. However, if the Critique pattern is a major structural focus of the segment design (as is the case with day 2 of the Hashing course segment), it is likely worth calling attention to its instantiation. A similar decision must be made with trend patterns, which come next in the annotation process. When the trend is important to the overall pedagogical success of the course segment, a trend annotation should be applied. This calls out the feature as significant and makes it accessible to novices inspecting the course.

Multi-step pattern annotations come next in the process. As each step of a multi-step pattern is fulfilled by a precise type of content, these patterns have well defined mappings to instantiations and relate directly to content tags. Thus, in curriculum that has been fully tagged using the tag library of a pattern language, the applicable patterns will be fairly obvious. Again, the choice of annotations becomes primarily a matter of what the annotator wishes to emphasize about a particular course segment. At the end of this process, a full pattern-annotated course segment has been produced and is ready for consumption by novices and validation by experts.

6.4 Discussion of Pattern Utility

It should be clear from the preceding discussion that some pedagogical patterns have far greater utility than others as annotations in a pattern-annotated course segment and for aiding in learning within the VDA model. In particular, only those patterns that can be reasonably translated into single-step, multi-step, and trend patterns can act as high-quality annotations. These are the patterns that are quintessentially generative – they describe learning patterns that are both instantiable (able to be translated into tangible curriculum designs) and general enough to be implemented in a variety of different ways in different contexts. The other pattern types fail on at least one of these criteria.

The advice patterns offer curriculum design suggestions that are not instantiable at the scope of design that I am interested in with PACT or VDA, a scope that ranges from the level of designing individual learning activities all the way up to the organization of all the activities that make up the full sequence of a course. Advice like "grade team projects fairly" or "have the teacher select teams rather than allowing students to self-select" are reasonable (specifically, I do not disagree with the intentions or rationale behind these patterns), but they do not operate at the content level and arguably are not design patterns at all. Regardless of whether one feels that advice like this has a place in the pedagogical design patterns genre, they clearly are of no utility to an instructor seeking to write next week's lab material.

The special patterns suffer from the opposite problem. They are over-specified and cannot be instantiated in any reasonable way other than exactly how they were written. They are effectively fully-developed exercises and assignments that can be dropped into a course without further thought. In this regard, they are very similar to the popular "nifty assignments" (Parlante, 2007) that are presented at each year's SIG-CSE Symposium. A repository of such learning modules (such as many of the existing digital library projects (Borgman, 1999; Schatz & Chen, 1996)) could be a very helpful thing, but I would argue that they are not, fundamentally, patterns. I have made an effort to segregate the special patterns from the more generative patterns within the PACT Annotation Library to avoid confusing novice users and leading them to believe that the two concepts are interchangeable.

What does this all mean for pedagogical pattern designers? Essentially, that there is a fine line that must be walked when developing a pattern language. If the patterns drift too far towards giving advice, rather than focusing on how the pattern can be applied to content design, then they will not be useful to content creators. Perhaps such quasi-patterns would be better suited for a list of "tips and tricks" rather than attempting to shoehorn them into the Alexander Pattern format. In the other direction, if the patterns become too specific then they are really no longer patterns at all, but fully realized learning modules. Such modules have a place, but I do not believe that it is in a pattern language. By focusing on the middle ground – the generative space that provides a pedagogical rationale that can be realized in content in many different ways – pedagogical patterns can become more useful and less confusing to the average reader.

Chapter 7

PACT Generation Two: An Overlay Curriculum Design Tool

The second version of the PACT tool was designed based upon the lessons learned from my initial PACT deployment and an improved understanding of the learning sciences principles that form the foundation of my work. An iterative design process was used to incrementally improve upon the initial PACT design, informed by continuous feedback from the UC-WISE research group. The resulting tool is a far more robust, production-quality application that can be used to review, design, iteratively improve, and learn from pattern annotated courses. Designed in tandem with the Virtual Design Apprenticeship framework described in Chapter 2, PACT Generation Two represents an effort to tightly couple the VDA principles with concrete realizations aimed at the space of curriculum design.

In this chapter, I discuss the design changes in PACT Generation Two (PACT2), focusing on their utility for novices in the process of learning about best practices in lab-centric instructional design. The utility of this new design is validated in Chapter 8, where I present a full user study showing that PACT2 has a substantial impact on how effectively novices learn from examples, how they perceive the credibility of examples, and the quality of curricular materials they can produce.

7.1 Revised PACT Design

PACT2 was designed from the ground up to be flexible and extensible. The essential aims were to make it easier for annotators to express their pedagogical intentions within the tool and to make the annotations created by experts more useful to novices. Based on feedback received during the deployment of PACT1, I arrived at the following target design goals for the redesign:

- To make PACT more useful to all designers (novices and experts alike) by integrating the tool with real-world learning management systems.
- To incorporate the PACT Annotation Library and PACT Annotation Schema (discussed in Chapter 6) by providing formalized, concrete annotations for pedagogical patterns and visual representations that make the connections between components of a pattern and the content that instantiates it more salient.
- To leverage formalized pattern annotations to provide basic feedback to users on whether or not they are instantiating patterns in the ways intended by the pedagogical experts who created them.
- To provide an intuitive and useful mechanism for using pattern annotations as templates for new design components.
- To reveal more information about patterns and the principles behind them within the tool.

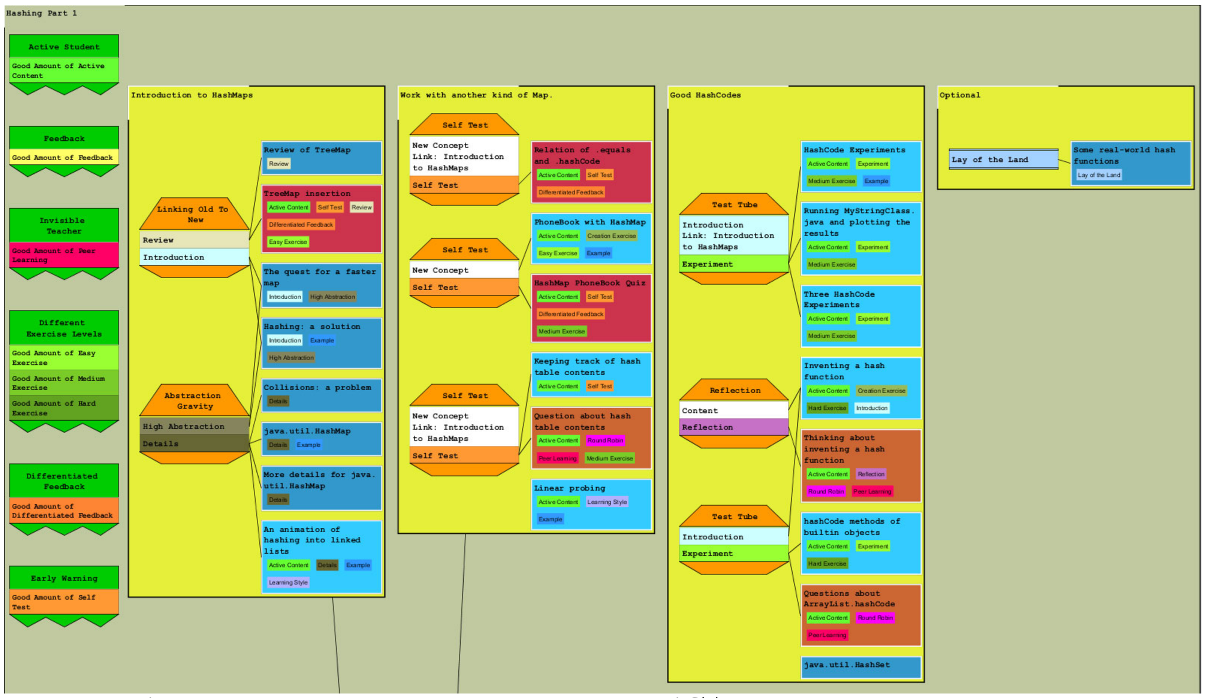

Hashing Part 1
Active Student
Good Amount of Active Content
Feedback
Good Amount of Feedback
Invisible Teacher
Good Amount of Peer Learning
Different Exercise Levels
Good Amount of Easy Exercise
Good Amount of Medium Exercise
Good Amount of Hard Exercise
Differentiated Feedback
Good Amount of Differentiated Feedback
Early Warning
Good Amount of Self Test
Introduction to HashMaps
Linking Old To New
Review
Introduction
Abstraction Gravity
High Abstraction
Details
Review of TreeMap
TreeMap insertion
The quest for a faster map
Hashing: a solution
Collisions: a problem
java.util.HashMap
More details for java.util.HashMap
An animation of hashing into linked lists
Work with another kind of Map.
Self Test
New Concept
Link: Introduction to HashMaps
Relation of .equals and .hashCode
PhoneBook with HashMap
HashMap PhoneBook Quiz
Keeping track of hash table contents
Question about hash table contents
Linear probing
Good HashCodes
Test Tube
Introduction
Link: Introduction to HashMaps
Experiment
Reflection
Content
HashCode Experiments
Running MyStringClass.java and plotting the results
Three HashCode Experiments
Inventing a hash function
Thinking about inventing a hash function
hashCode methods of builtin objects
Questions about ArrayList.hashCode
java.util.HashSet
Optional
Lay of the Land
Some real-world hash functions

Figure 7.1: A single day of annotated content displayed in the PACT2 interface. Large yellow boxes are groups of learning activities. Small rectangles are the activities, colored based on their UC-WISE step types. The small colorful labels within each activity are tags. The remaining shapes are pattern annotations, with six trend annotations on the far left, eight multi-step annotations throughout the center, and one single-step annotation on the far right.

I achieved these goals through a comprehensive redesign of the PACT system. While most aspects of the design changed in some way between PACT1 and PACT2, the high-level look and feel of the editor remained similar. The PACT2 course view remains a zoomable, direct manipulation interface that allows users to quickly reorganize, create, and delete course contents and annotations. Figure 7.1 shows a single day of annotated content from CS61BL in the PACT2 interface and Figure 7.3 shows a close-up of a single activity. The major changes in design within the PACT2 interface, and how they were constructed to realize VDA's design principles, are described in the remainder of this section.

7.1.1 *PACT as an Overlay Design Tool*

PACT2 is implemented as an overlay curriculum design tool rather than as a standalone tool. That is, PACT2 supplies a new design interface that can be attached to existing learning management systems (LMS). PACT2 synchronizes with the LMS via the LMS's existing APIs to build a shared model of the contents of a course (Figure 7.2). The PACT and LMS models are synchronized by frequent

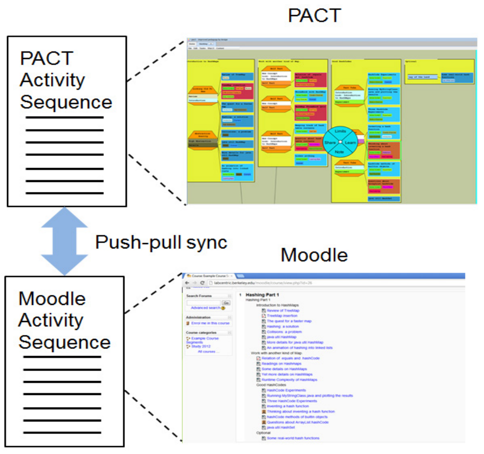


Figure 7.2: PACT running as an overlay design tool on top of an existing course authoring tool, Moodle.

push-pull requests by PACT2. Changes to the course model in either tool will quickly be reflected in the other. Thus, PACT2 can take advantage of the features that LMS typically do well (creating individual learning activities and presenting them to students) while providing the curriculum author with tools to manage the complexity of high-level design of a full course.

The implications of this change are significant. Fundamentally, it allows PACT2 to integrate into the real-world workflows of both experts and novices, fulfilling the VDA design principle to **make design artifacts usable in the real world**. Experts can work in PACT2 while doing authentic design work (intended to be delivered in their actual classrooms), building annotations in step with the content creation process. Novices can learn about best practices in learner-centered curriculum design while also creating a real course – one that can be used by real students with no additional steps or effort required on the part of the user.

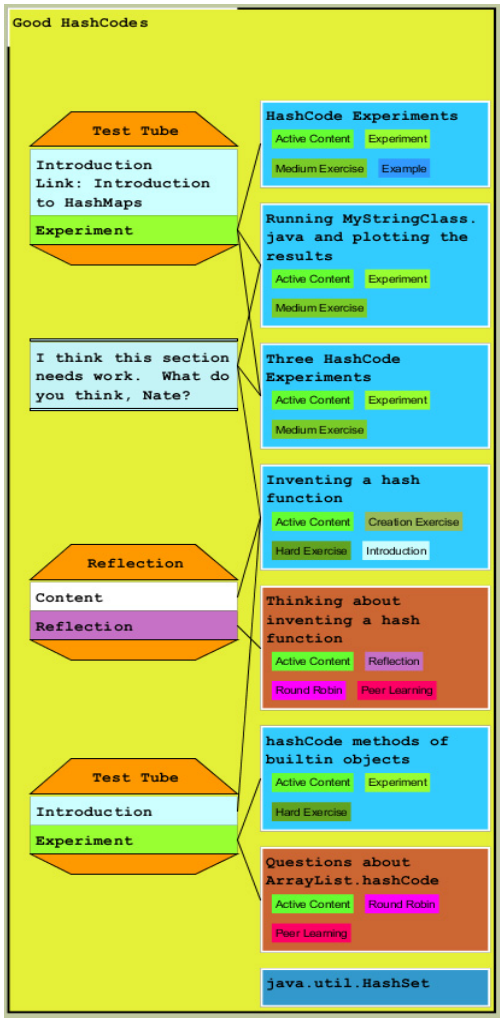


Figure 7.3: A single group of activities with annotations. The left column is the annotations. The right column is the individual activities. Test Tube and Reflection are multi-step patterns. Lines link the stages of a pattern to the activities that instantiate it.

7.1.2 *Formalized PAS Annotations*

The new PACT2 interface is deeply tied to the more robust model of pedagogical patterns and their instantiations that is provided by the PACT Annotation Schema. Where PACT1 implemented pattern annotations as simple notes – undefined and underspecified blobs that could be attached in free-form ways to content – PACT2 uses highly structured visual representations of patterns in its annotations. These structured annotations display exactly how each pattern works in a clear manner that can be understood at a glance.

Each type of pattern annotation specified in PAS (as described in Chapter 6) has its own unique visual representation in PACT2. Tags are shown as colored rectangles drawn directly on top of the tagged content (Figure 7.3). Related tags are given similar colors, allowing the user to discern at a glance just what each piece of content is and what its role is within the curriculum. Multi-step pattern annotations are represented by geometric shapes with "slots" for each step in the pattern (Figure 7.4). These slots can be linked directly to content in the design, which is shown with a line bridging the two elements. Each slot is colored and labeled according to the type of content (the

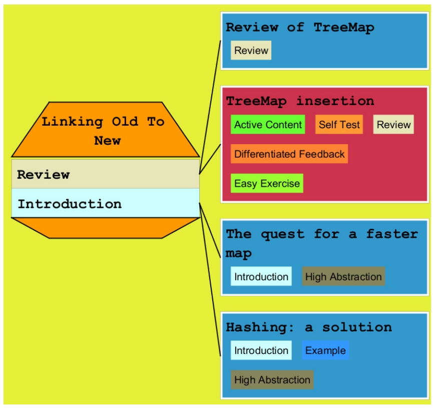


Figure 7.4: A multi-step pattern annotation. These types of patterns have a sequence of steps that must be completed in order to instantiate the pattern. Each step is satisfied by one or more content types, as indicated by the text and color within the step. These steps need to be connected to content of the appropriate type, or else they will signal a warning.

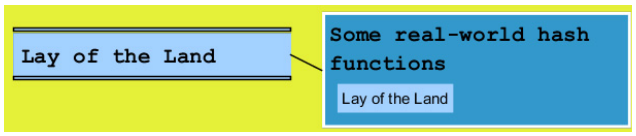


Figure 7.5: Single-step patterns are patterns that can be instantiated with a single activity. In these cases, the pattern is simply indicated by placing a tag on the piece of content. If the author would like to emphasize the single-step pattern they can also create a call-out annotation, as seen here.

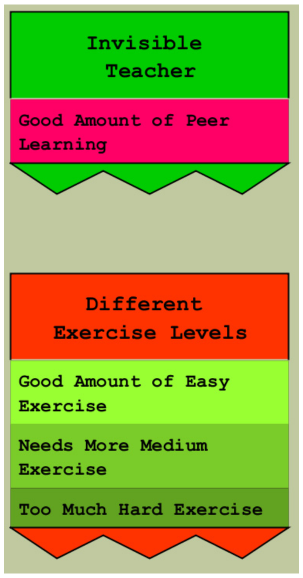

Invisible Teacher
Good Amount of Peer Learning
Different Exercise Levels
Good Amount of Easy Exercise
Needs More Medium Exercise
Too Much Hard Exercise

Figure 7.6: A trend pattern that has been correctly instantiated (Invisible Teacher) and one that has not been correctly instantiated (Different Exercise Levels).

content tag or tags) that is required to properly instantiate it. Single-step pattern annotations are visualized as rectangular callouts colored to match their corresponding content type, reflecting that these patterns are simply promotions of tags up to a higher level of visual prominence (Figure 7.5). Trend pattern annotations in PACT2 are placed on to higher-level containers, such as whole days, weeks, or even entire semesters of content (Figure 7.6). These annotations are also colored to match the types of content intended to instantiate the pattern specification and serve as a constant visual reminder to the user that expert designers include specific amounts of the relevant types of content. Finally, special and advice patterns, which have no deeply-seated structural ties to content in their specifications, are represented in PACT2 as general note annotations (like the freeform comment in Figure 7.3 and similar to how all annotations were presented in PACT1).

These changes serve to make the relationships between annotations and content much more obvious – able to be taken in with a quick inspection of the course view. This is a realization of the VDA principle to **make annotated artifact exploration and comprehension easy**, and is far more comprehensive than just the intuitive navigation features that partially fulfilled this principle in the PACT1 design.

7.1.3 Pattern Constraint Checking

Having a formal model for representing pedagogical patterns provides PACT2 with opportunities for interface improvements beyond simply improving the visual display. I also leverage the pattern constraint system of PAS to provide feedback to users on how accurately their instantiations of pedagogical patterns match the underlying pedagogical principles of the pattern. PACT2 continuously monitors the course design while the user is working and automatically warns the user when an edit to the course violates pattern constraints. These warnings come in the form of gentle but obvious changes to the interface that signal a problem. The user can then open more information about the problem and learn how to fix it. These features build on top of the informal interface features of PACT1 to further realize the VDA principle of **encourage design experimentation, but preserve expert design intent**.

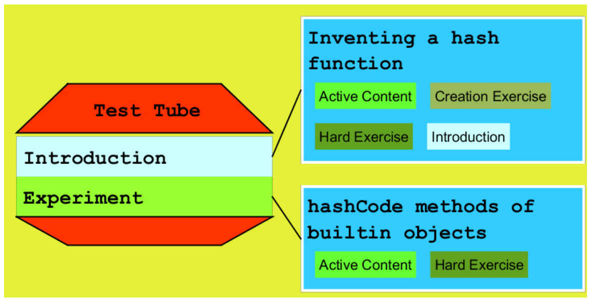


Figure 7.7: This pattern has not been correctly instantiated, because its second stage expects to be connected to an Experiment. The display signals this error by turning red.

These constraint satisfaction warnings come in four flavors in PACT2, corresponding to the four types of constraints in PAS. First, PACT2 ensures that multi-step and single-step pattern annotations are connected to content types that match their specification. When this is not the case, the annotation turns red (Figure 7.7) and the step or steps that have been mismatched to content change text to indicate the issue. Second, trend pattern annotations in PACT2 can signal warnings when the trend indicated by the pattern is not being followed. They do this by, again, turning red on the trim and changing text to indicate that there is “not enough” or “too much” of a given content type within the annotated curriculum container (Figure 7.6). Finally, multi-step annotations can warn the user when temporal constraints are not satisfied – that is, when components of the instantiated pattern are either too close to or too far away from each other. This is done by changing the color of the offending pattern annotation and thickening the link between the annotation and the offending piece of content.

7.1.4 Using Pattern Annotations as Design Templates

The PAS formalization of patterns into annotations opens the door for one more significant interface innovation in PACT2. Once annotated, existing curriculum designs can be used as templates for new designs. If an instructor maintains the pedagogical intent (activity types, tags, and pattern annotations) of a good design he should be able to change the content of that design to fit a new context while preserving the pedagogy of the original design (this premise is tested and validated in the user study described later in this chapter). This common task is supported by annotation cloning in PACT2. A user can select a piece of curriculum and then generate a copy of it that retains design annotations but has been stripped of all specific content (Figure 7.8). Thus, PACT2 instantiates the VDA principle to **use annotations as templates for new designs**.

The pattern cloning mechanism also touches on a fundamental question about the PACT design in general: how are pattern annotations introduced to designs being created by novices when they do not understand the design space well enough to deal with a full library of pattern annotations? The answer is that pattern annotations come into a novice user's design artifact by having been cloned or copied from an expert's design. In practice, this feature has proven to be tremendously useful for exactly this reason – it helps novice designers structure their thoughts around the scaffolding provided by expert annotations while allowing the user to build a course from scratch, not simply modifying an existing course *in situ*.

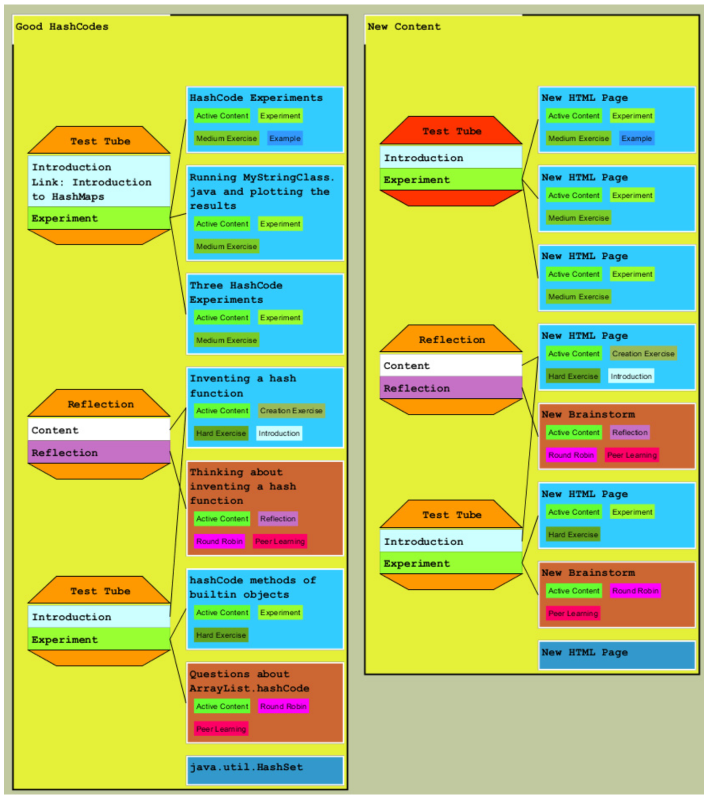


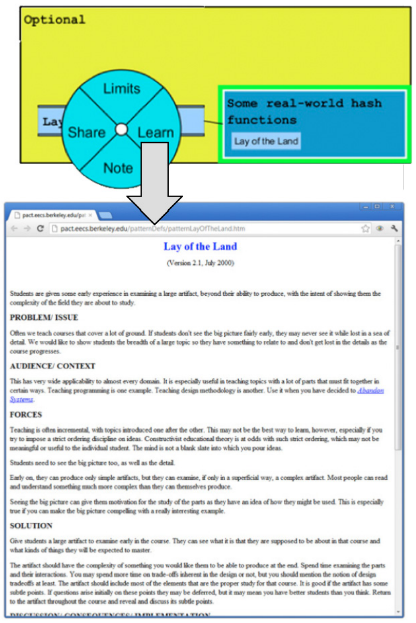
Figure 7.8: The user has used the Clone feature to create an empty copy of an activity container. The annotations and activity types are preserved into the new activity, but all contents (titles, text, student data, etc.) have been stripped out. This is useful for creating new course materials that follow a successful template.

7.1.5 Revealing Pattern & Tag Details On Demand

Pattern annotations in PACT1 provided only shallow ties to the underlying pedagogical patterns of an annotation. Essentially, PACT1 revealed to the user the name of each pattern involved in an annotation (in course view), but nothing more. Interested individuals would have to track down that reference and read the full pattern document to learn more about the pattern.

In PACT2, I have integrated the details of patterns directly into the interface. As discussed in Chapter 6, I produced a short summary of each pattern and tag included in the PACT Annotation Library. These summaries are embedded in the PACT2 display so that interested users can peruse them. Further, if a user wants to move beyond summaries of the pedagogical patterns, the original source material for each pattern can be opened in a web browser via a simple menu selection in PACT2 (Figure 7.9). In this way, PACT2 realizes the VDA principle to **link annotations directly to further reading**.

It is desirable, however, to hide these details from users most of the time; the precise details of a pedagogical pattern are only sometimes relevant to the task of building new courses or learning about expert designs. Hiding the details avoids cluttering the display and helps keep novice users from feeling overwhelmed by abstract details they are not yet ready to deal with. For these reasons, pattern details are produced only on-demand within the PACT2 interface. To see them, the user simply hovers their mouse cursor over any tagged content or pattern annotation – the details then appear in a tooltip-style popup.


Optional
Limits
Lay
Share
Learn
Note
Some real-world hash functions
Lay of the Land
pact.eecs.berkeley.edu/pa
pact.eecs.berkeley.edu/patternDefs/patternLayOfTheLand.htm

Lay of the Land

(Version 2.1, July 2000)

Students are given some early experience in examining a large artifact, beyond their ability to produce, with the intent of showing them the complexity of the field they are about to study.

PROBLEM/ ISSUE

Often we teach courses that cover a lot of ground. If students don't see the big picture fairly early, they may never see it while lost in a sea of detail. We would like to show students the breadth of a large topic so they have something to relate to and don't get lost in the details as the course progresses.

AUDIENCE/ CONTEXT

This has very wide applicability to almost every domain. It is especially useful in teaching topics with a lot of parts that must fit together in certain ways. Teaching programming is one example. Teaching design methodology is another. Use it when you have decided to *Abandon Systems*.

FORCES

Teaching is often incremental, with topics introduced one after the other. This may not be the best way to learn, however, especially if you try to impose a strict ordering discipline on ideas. Constructivist educational theory is at odds with such strict ordering, which may not be meaningful or useful to the individual student. The mind is not a blank slate into which you pour ideas.

Students need to see the big picture too, as well as the detail.

Early on, they can produce only simple artifacts, but they can examine, if only in a superficial way, a complex artifact. Most people can read and understand something much more complex than they can themselves produce.

Seeing the big picture can give them motivation for the study of the parts as they have an idea of how they might be used. This is especially true if you can make the big picture compelling with a really interesting example.

SOLUTION

Give students a large artifact to examine early in the course. They can see what it is that they are supposed to be about in that course and what kinds of things they will be expected to master.

The artifact should have the complexity of something you would like them to be able to produce at the end. Spend time examining the parts and their interactions. You may spend more time on trade-offs inherent in the design or not, but you should mention the notion of design tradeoffs at least. The artifact should include most of the elements that are the proper study for that course. It is good if the artifact has some subtle points. If questions arise initially on these points they may be deferred, but it may mean you have better students than you think. Return to the artifact throughout the course and reveal and discuss its subtle points.

DISCUSSION/ CONSEQUENCES/ IMPLEMENTATION

Figure 7.9: The user has opened the annotation menu and selected the Learn option. This opens the full text of the pattern referenced by this annotation in a web browser.

7.2 PACT2 Implementation Details

Like PACT1, PACT2 is written in the Java programming language and is primarily based upon the Piccolo zoomable user interface library. PACT2 also uses XML files as its primary data persistence mechanism. This, however, is where the similarities to PACT1 end. Where the initial implementation of PACT was very rigid and difficult to extend, PACT2 was built from the ground up as an extensible system with a core engine that makes decisions based on input from a number of small plugins. These plugins may represent new presentation views (e.g. a view of a course that highlights learning activities that students have complained about in the past), links to outside systems (e.g. the publishing plugin that synchronizes PACT with Moodle), collections of annotations (e.g. the PACT Annotation Library is built as a collection of plugins, one for each source document in the Pedagogical Patterns Project), or any number of other things. Each of these plugins may influence the way that one or more components in the core PACT2 engine processes data and makes decisions. For instance, the layout engine uses a force-based algorithm (Fruchterman & Reingold, 1991) that asks each plugin to state its own preferences for where each item in the display should be placed and how they should be sized. It then builds a consensus opinion from all of the plugins (weighted based on knowledge of the user's current task) to arrive at the final layout. This system allows PACT2 to be easily used in ways that were not anticipated in the original design.

The full source of PACT2, excluding external libraries, is approximately 20,000 lines of code.

Figures 7.10 – 7.16 show the example course segments that I produced to demonstrate the utility of the PACT Annotation Library (discussed in Section 6.3) as displayed in PACT. The Hashing, Inheritance, and Recursion segments are shown fully annotated. The Trees segment is shown with no annotations, as a point of comparison.

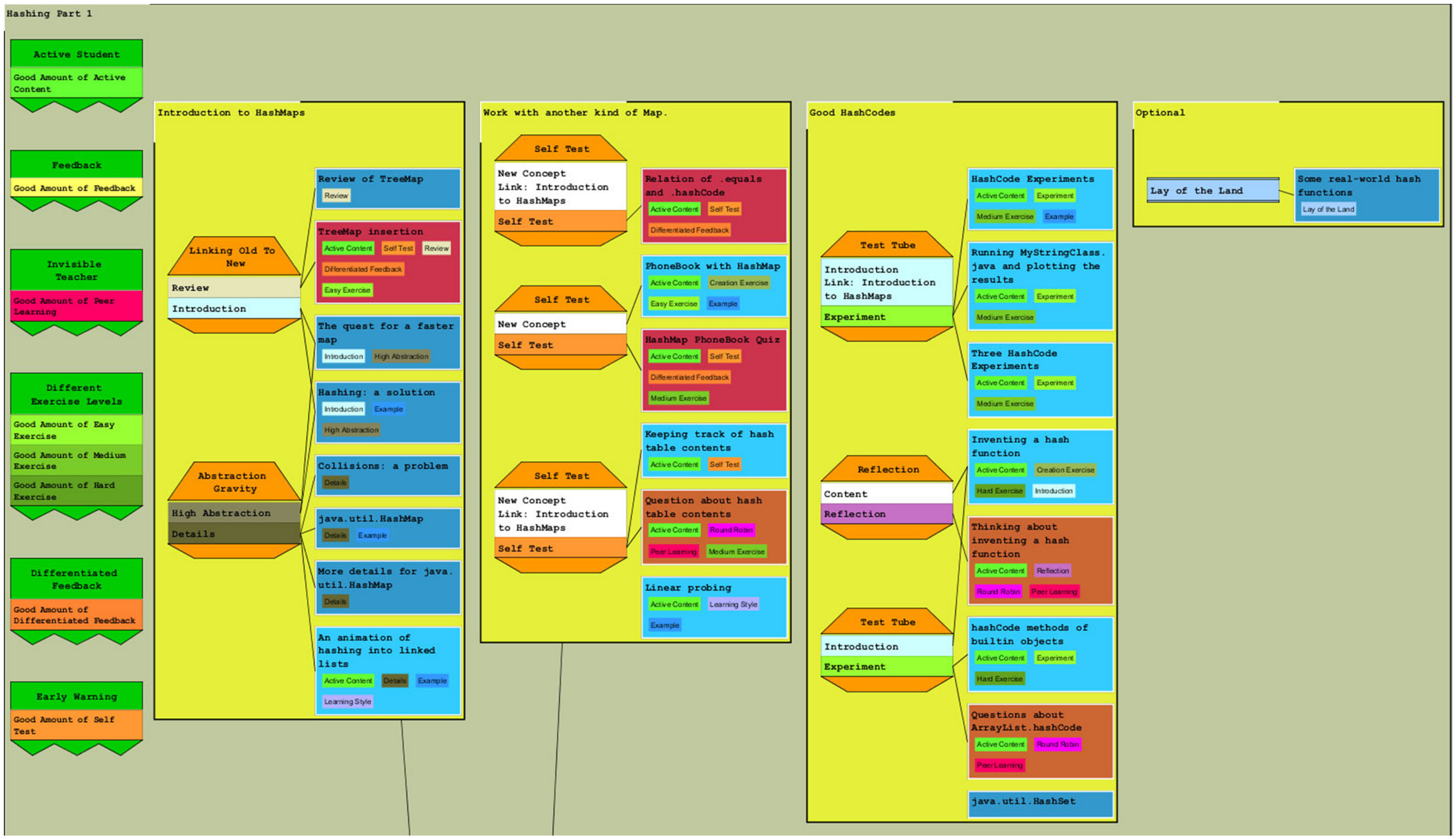


Figure 7.10: Day 1 of the Hashing course segment from CS61bl, annotated in PACT.

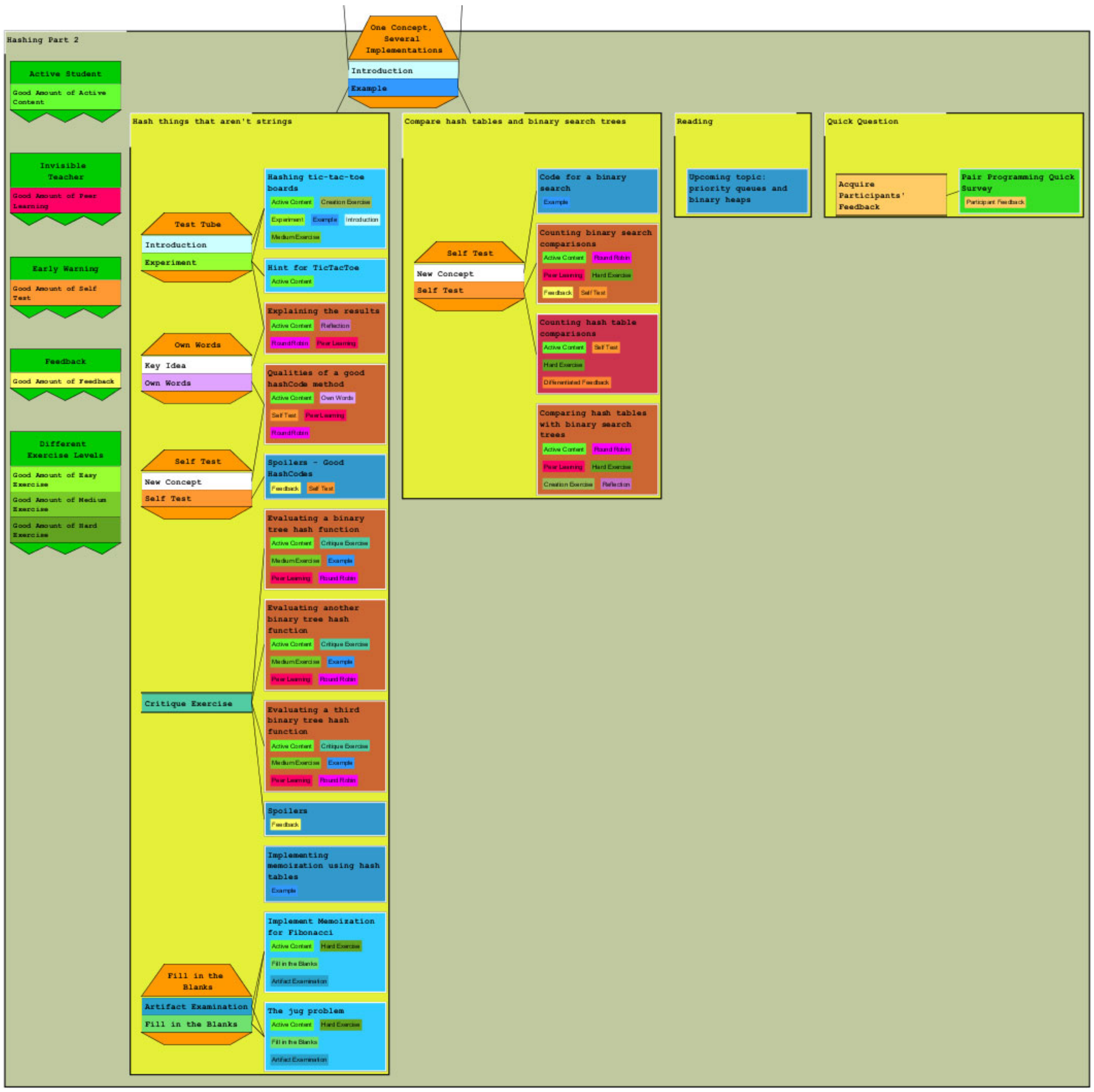


Figure 7.11: Day 2 of the Hashing course segment from CS61bl, annotated in PACT.

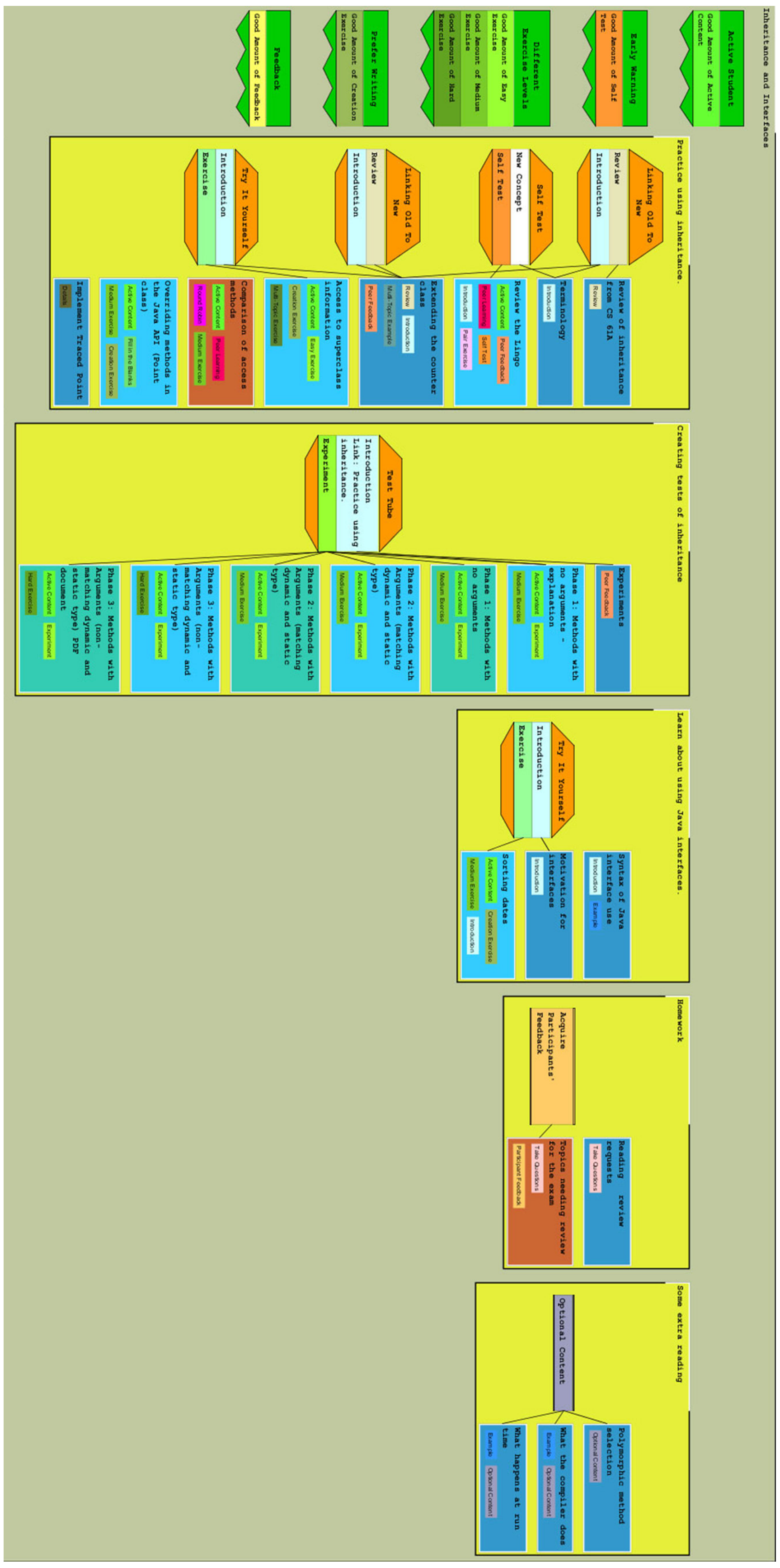


Figure 7.12: Day 1 of the Inheritance course segment from CS61bl, annotated in PACT

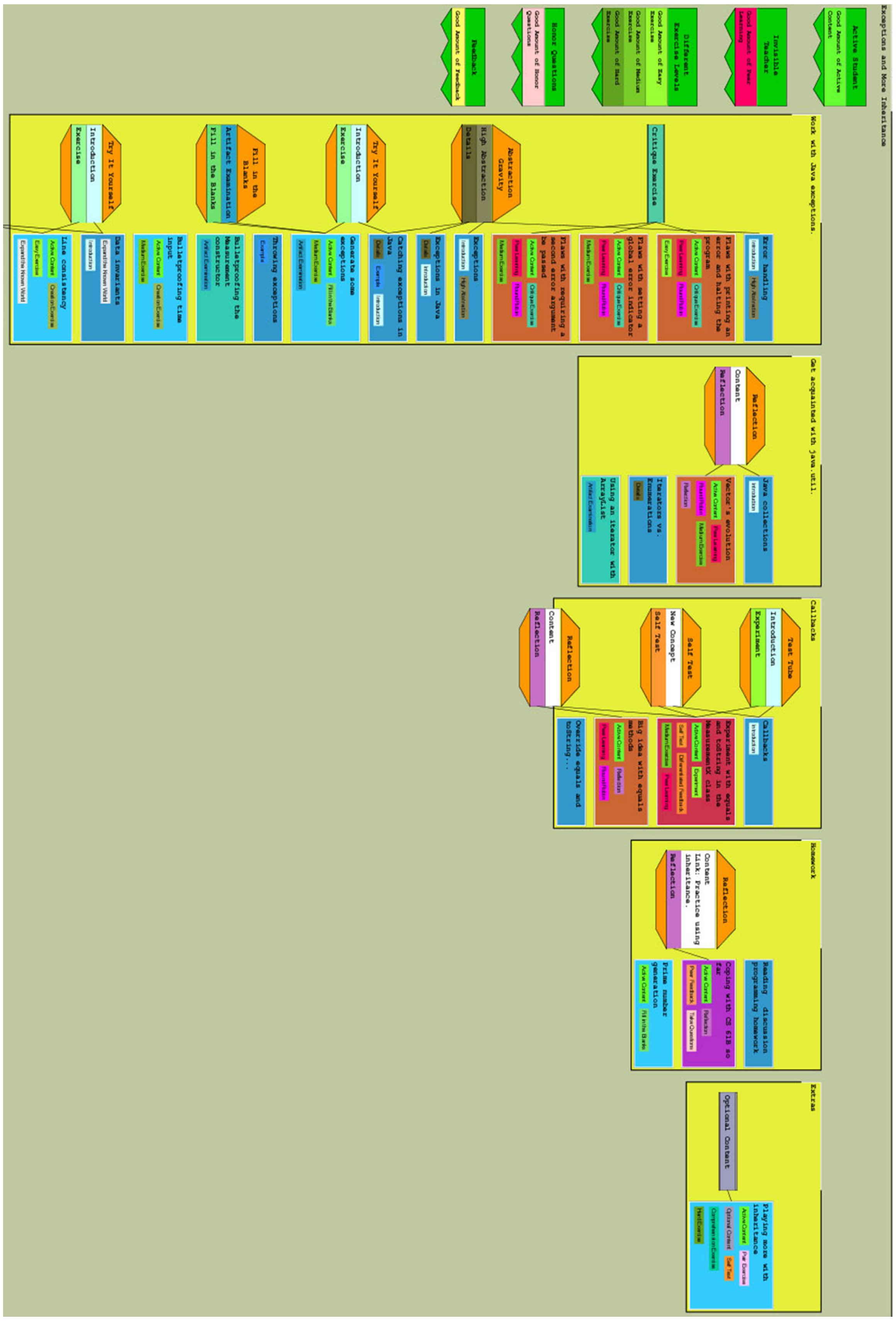

Exceptions and More Inheritance
Active Student
Good Amount of Active Content
Invisible Teacher
Good Amount of Peer Learning
Different Exercise Levels
Good Amount of Easy Exercise
Good Amount of Medium Exercise
Good Amount of Hard Exercise
Honor Questions
Good Amount of Honor Questions
Feedback
Good Amount of Feedback
Work with Java exceptions.
Critique Exercise
Abstraction Gravity
High Abstraction
Details
Try It Yourself
Introduction
Exercise
Fill in the Blanks
Artifact Examination
Fill in the Blanks
Try It Yourself
Introduction
Exercise
Error handling
Flaws with printing an error and halting the program
Flaws with setting a global error indicator
Flaws with requiring a second error argument be passed
Exceptions
Exceptions in Java
Catching exceptions in Java
Generate some exceptions
Throwing exceptions
Bulletproofing the Measurement constructor
Bulletproofing time input
Data invariants
Line consistency
Get acquainted with java.util.
Reflection
Content
Reflection
Java collections
Vector's evolution
Iterators vs. Enumerations
Using an iterator with ArrayList
Callbacks
Test Tube
Introduction
Experiment
Self Test
New Concept
Self Test
Reflection
Content
Reflection
Callbacks
Experiment with equals and toString in the Measurement class
Big idea with equals methods
Override equals and toString...
Homework
Reflection
Content
Link: Practice using inheritance.
Reflection
Reading discussion programming homework
Coping with CS 61B so far
Prime number generation
Extras
Optional Content
Playing more with inheritance

Figure 7.13: Day 2 of the Inheritance course segment from CS61bl, annotated in PACT.

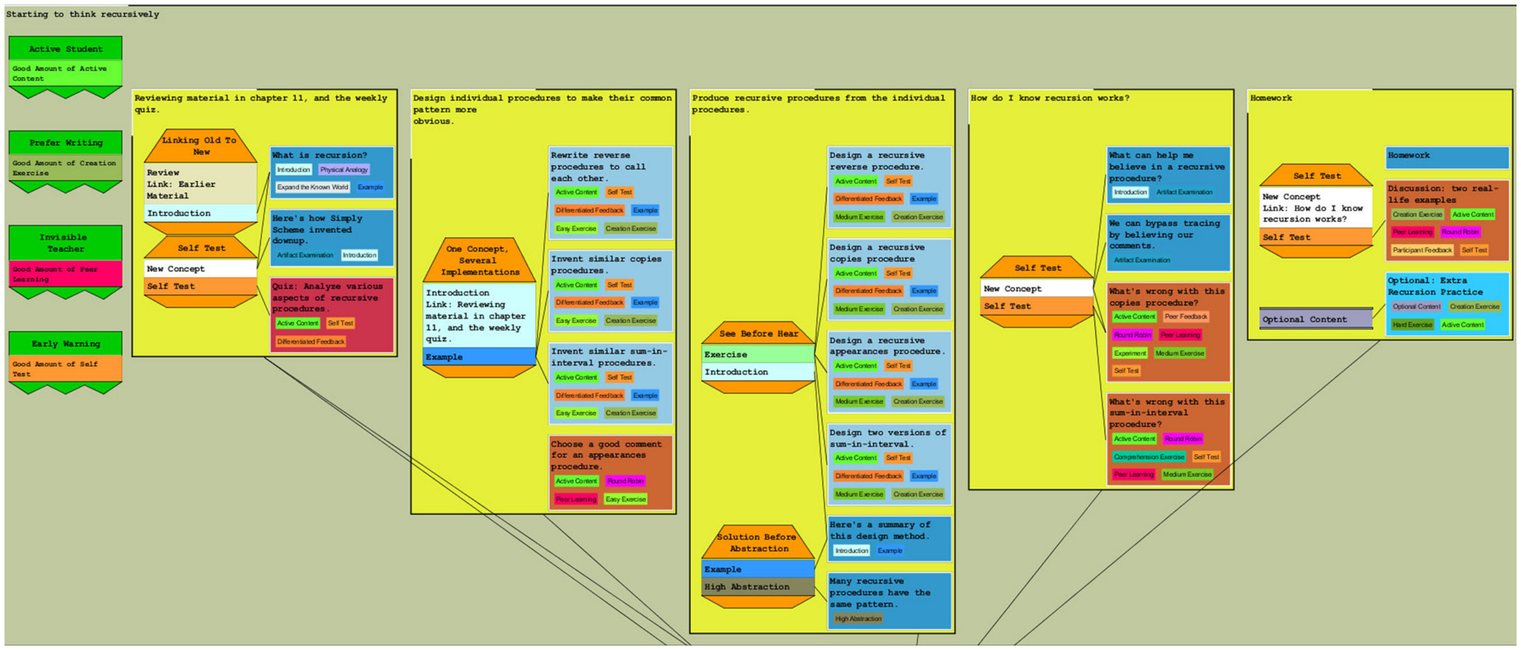


Figure 7.14: Day 1 of the Recursion course segment from CS3l, annotated in PACT.

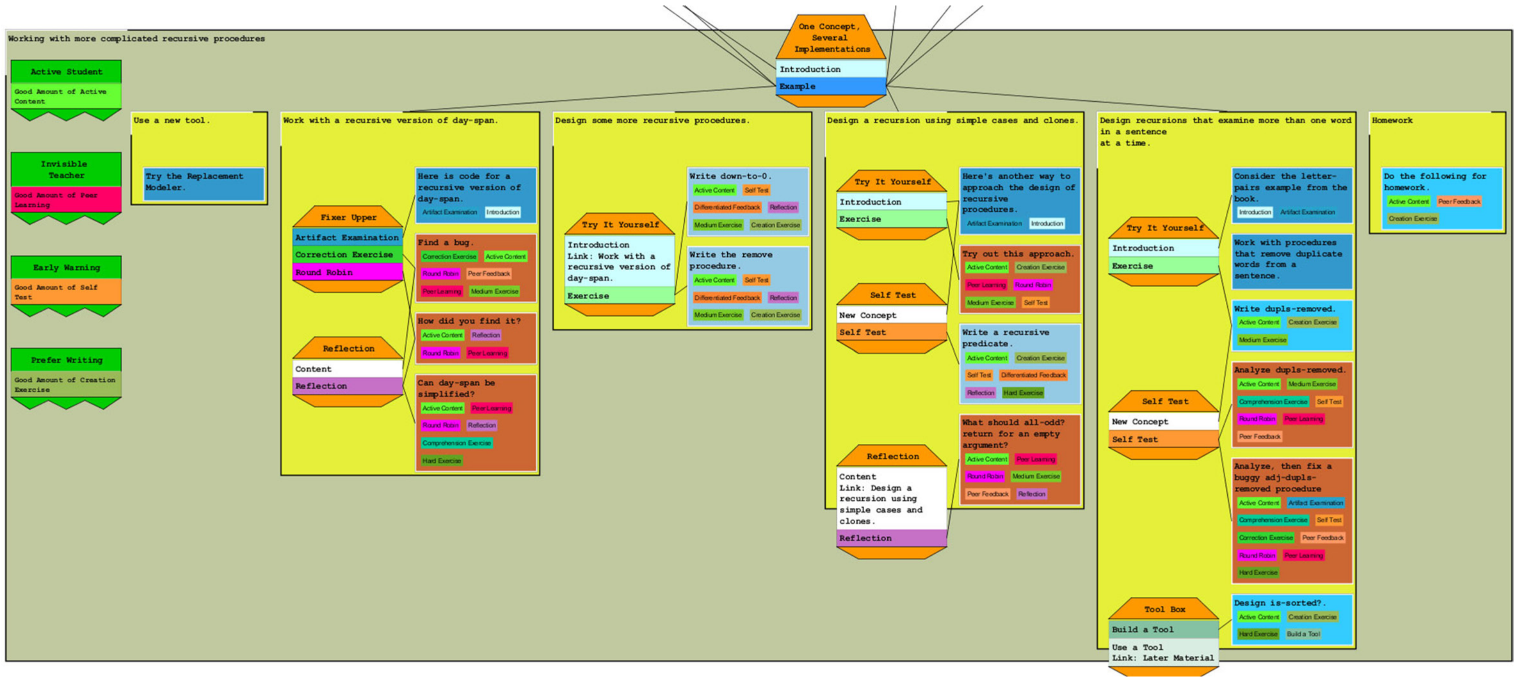

One Concept, Several Implementations
Introduction
Example
Working with more complicated recursive procedures
Active Student
Good Amount of Active Content
Invisible Teacher
Good Amount of Peer Learning
Early Warning
Good Amount of Self Test
Prefer Writing
Good Amount of Creation Exercise
Use a new tool.
Try the Replacement Modeler.
Work with a recursive version of day-span.
Fixer Upper
Artifact Examination
Correction Exercise
Round Robin
Reflection
Content
Reflection
Here is code for a recursive version of day-span.
Find a bug.
How did you find it?
Can day-span be simplified?
Design some more recursive procedures.
Try It Yourself
Introduction Link: Work with a recursive version of day-span.
Exercise
Write down-to-0.
Write the remove procedure.
Design a recursion using simple cases and clones.
Try It Yourself
Introduction
Exercise
Self Test
New Concept
Self Test
Reflection
Content Link: Design a recursion using simple cases and clones.
Reflection
Here's another way to approach the design of recursive procedures.
Try out this approach.
Write a recursive predicate.
What should all-odd? return for an empty argument?
Design recursions that examine more than one word in a sentence at a time.
Try It Yourself
Introduction
Exercise
Self Test
New Concept
Self Test
Tool Box
Build a Tool
Use a Tool Link: Later Material
Consider the letter-pairs example from the book.
Work with procedures that remove duplicate words from a sentence.
Write dupls-removed.
Analyze dupls-removed.
Analyze, then fix a buggy adj-dupls-removed procedure
Design is-sorted?.
Homework
Do the following for homework.
Active Content
Creation Exercise
Peer Feedback
Self Test
Differentiated Feedback
Medium Exercise
Hard Exercise
Artifact Examination
Introduction
Correction Exercise
Round Robin
Peer Learning
Comprehension Exercise
Build a Tool

Figure 7.15: Day 2 of the Recursion course segment from CS3l, annotated in PACT.

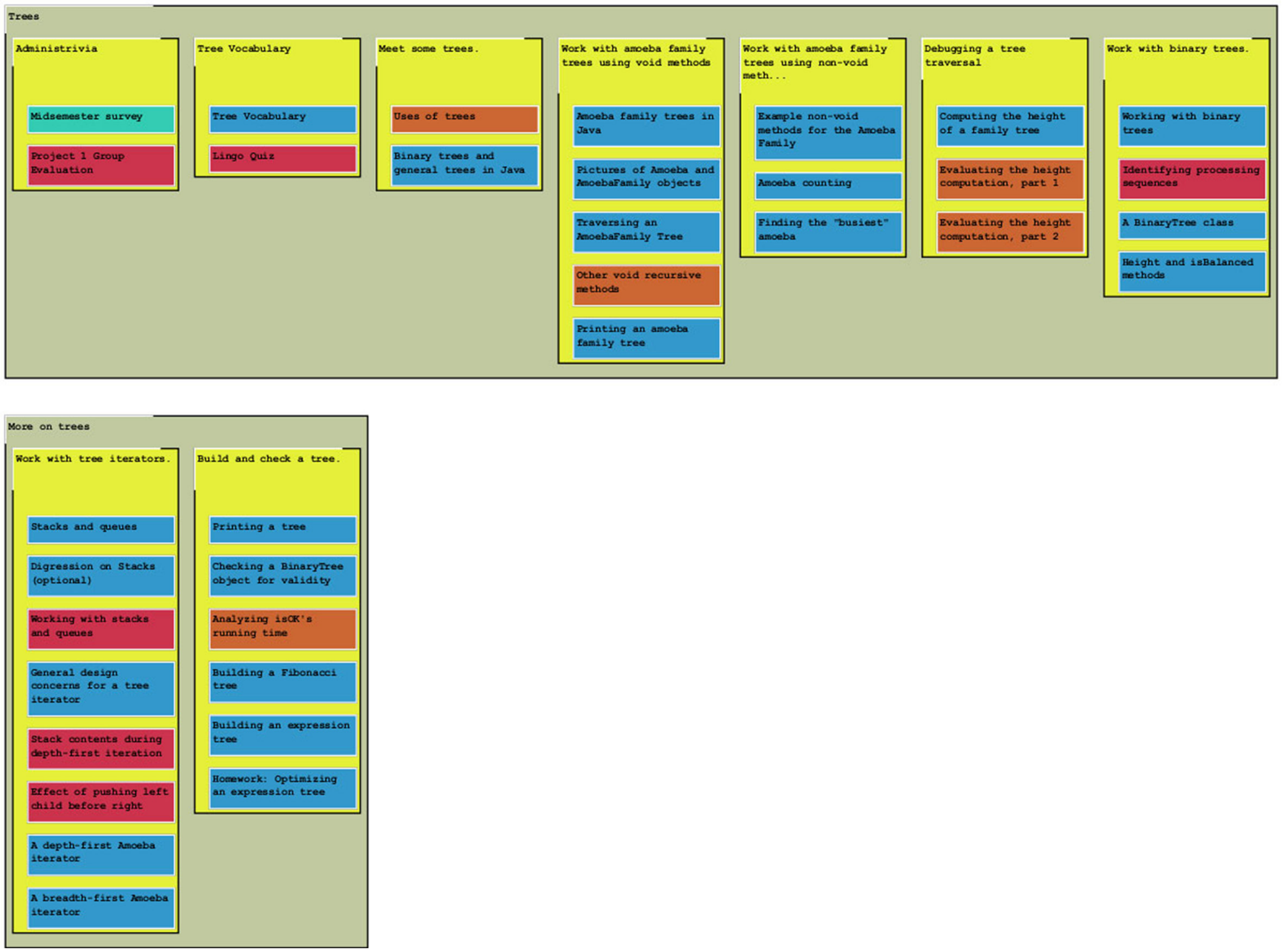

Trees
Administrivia
Midsemester survey
Project 1 Group Evaluation
Tree Vocabulary
Tree Vocabulary
Lingo Quiz
Meet some trees.
Uses of trees
Binary trees and general trees in Java
Work with amoeba family trees using void methods
Amoeba family trees in Java
Pictures of Amoeba and AmoebaFamily objects
Traversing an AmoebaFamily Tree
Other void recursive methods
Printing an amoeba family tree
Work with amoeba family trees using non-void meth...
Example non-void methods for the Amoeba Family
Amoeba counting
Finding the "busiest" amoeba
Debugging a tree traversal
Computing the height of a family tree
Evaluating the height computation, part 1
Evaluating the height computation, part 2
Work with binary trees.
Working with binary trees
Identifying processing sequences
A BinaryTree class
Height and isBalanced methods
More on trees
Work with tree iterators.
Stacks and queues
Digression on Stacks (optional)
Working with stacks and queues
General design concerns for a tree iterator
Stack contents during depth-first iteration
Effect of pushing left child before right
A depth-first Amoeba iterator
A breadth-first Amoeba iterator
Build and check a tree.
Printing a tree
Checking a BinaryTree object for validity
Analyzing isOK's running time
Building a Fibonacci tree
Building an expression tree
Homework: Optimizing an expression tree

Figure 7.16: The Trees course segment from CS61bl with no annotations in PACT.

Chapter 8

Evaluating PACT with Novice Designers: Learning, Perception, and Utility

After developing PACT2, I shifted my focus away from studying the activities of experts (as in the PACT1 contextual studies presented in Chapter 5) and began investigating PACT2 as used by novice designers. The primary activity of this phase of my research was designing and conducting a user study to examine how PACT's design and features impact the design practice of novices. The three main goals of this study were to investigate how PACT

- affects the way that novices learn from example curriculum designs,
- impacts novice users' perceptions of examples,
- and improves the quality of developed curricula.

8.1 Methodology

I recruited 26 graduate students from various engineering disciplines (e.g. computer science, mechanical engineering, industrial engineering) to participate in a lab study. While my intended audience for PACT is university-level instructors, as potential future faculty members graduate students are a reasonable proxy. Each of my participants had at least some teaching experience (generally far exceeding the UC Berkeley requirements to graduate with a PhD) and there is typically no required professional development between graduate school and starting a faculty position.

The study used a between-subjects design and participants were randomly assigned to one of two conditions: a treatment condition in which participants used PACT and a control condition in which participants used Moodle, the learning management system currently used to design and deliver lab-centric courses. As my randomization method ensured an even distribution between the two groups, each condition had 13 participants. Each study session was approximately 3 hours long and consisted of a background survey, a tutorial on LCI and the design tool, a design task, a post-task survey, and a summary interview. The evaluable outcomes of each of these components of the study are summarized in Table 8.1.

8.1.1 Background Survey

The first portion of the study was a background survey to gauge general information about the participant's relevant prior experience. It covered teaching experience, formal training as a teacher, attitudes towards teaching, familiarity with lab-centric instruction (and other computer-mediated instruction methods), and programming experience.

Background Survey	Teaching experience, attitudes on teaching, computer-mediated instruction background.
Design Task	One week (two lab sessions) of lab-centric curriculum.
Closing Survey	Likert scale responses on user experience and affect questions, summaries of the pedagogical approaches used in each example, measures of example persuasion.
Summary Interview	General comments on the design tools, task, and process.

Table 8.1: Evaluable outcomes from each stage of the study.

8.1.2 LCI & Design Tool Tutorial

The second stage of the study was a brief (approximately 10 minutes) tutorial on lab-centric instruction and how to use PACT or Moodle, per condition. Participants in both conditions were given an identical description of LCI and how it is used in the computer science lower division at UC Berkeley. This discussion included the strengths and weaknesses of LCI as an educational approach, practical details of how LCI labs are run, and a detailed description of each of the content types available within a LCI course. Details provided on the content types included specifics of how they are typically used in real courses and their intended pedagogical benefits.

Participants were shown how to use the design tool to create, edit, delete, and reorder content. I demonstrated browsing the outline view in both PACT and Moodle along with digging into the details of individual activities. In the PACT condition, participants were also shown how to create content via annotation cloning. Annotations and tags were briefly explained to PACT users so that they would understand how to interpret the display of example course materials (described below). Users were left with a written version of the tutorial that explained the available content types; the iconography used to represent those types within Moodle and the corresponding color-coding used to represent them within PACT; and instructions for operating the relevant design tool.

8.1.3 Curriculum Design Task

The largest portion of the user study was a design task, crafted to engage the participant in the design and development process of creating new curricula. Users in the PACT condition designed and developed their curricula in PACT while those in the Moodle condition used Moodle.

The design task was to develop one week (two three-hour sessions) of lab-centric material. Participants were asked only to create a high-level outline of their course segment, building a sequence of activities with descriptive titles, activity types (e.g. HTML page, Brainstorm, Quiz), and a brief summary of the intended contents of each activity. This reflects a common practice among expert curriculum designers, who often build up the sequence of activities before fleshing out the details. Participants were given up to two hours to complete the task. This limit was chosen for practical reasons, but also to reflect the breakneck pace at which curriculum design work is done in the real world – my contextual studies revealed that practicing instructors often develop both the outline and full contents of a lab day in two to four hours.

The subject to be covered by the course segment was an introduction to the C programming language. To mitigate the effects of prior knowledge of C, each participant was provided with materials from UC Berkeley's CS 61c, a lecture-based course that covered all the relevant topics. These materials consisted of one week (three lectures) of lecture slides and the two homework assignments from that week. Thus, the task was essentially about *translating* an already high-quality collection of lecture material into appropriate lab-centric material. This task mirrors the common real-world situation in which a subject-matter expert attempts to translate her expertise into a lab-centric curriculum.

8.1.3.1 Example Course Segments

The provided lecture materials serve as an example of the content to be covered, but provide no assistance with respect to making a proper lab-centric course. To help the participants make high-quality lab-centric segments, they were provided with four examples from real lab-centric computer science courses. Each example was of the same scope as the segment the participants were asked to create themselves: one week (two lab sessions) of lab-centric curricula. The specific pedagogical techniques used varied among the four examples, but they were fundamentally similar and all very carefully designed to exemplify best practices in LCI. All four examples included a good mix of active and passive content, numerous chances for the students to test their understanding, opportunities for reflection, and a broad mix of coding and comprehension exercises.

Three of the example segments were derived from UC Berkeley's CS61BL, an introductory Java-based data structures course, and covered Inheritance, Hashing, and Trees respectively. A fourth was derived from UC Berkeley's CS3L, a programming for non-majors course in Scheme, and covered Recursion. These examples were presented to the participants as having been contributed by experts in lab-centric instruction who were interested in helping the participant learn about best practices in LCI curriculum design. Participants were told that they could refer to the examples as much or as little as they found helpful in completing the design task – they were not instructed to learn anything from the examples beyond what was useful for the task at hand.

The presentation format of the examples was substantially different between the two experimental groups. The Moodle group was given all four examples as Moodle courses. They could view an outline of the materials, browse through the individual activities, and try active activities (e.g. quizzes) on their own all using the Moodle interface. The PACT group was given the four examples within PACT. They could browse the course outline within PACT and open individual activities for inspection (and to try) within the Moodle backend (as is done with PACT in actual practice).

The four examples presented in the PACT condition had varying degrees of annotation. The Inheritance and Hashing examples were highly annotated, with numerous pattern annotations and thorough tagging of individual activities. The Recursion example was lightly annotated, with high-level pedagogical patterns identified and the related tags applied to content, but without the level of annotation detail provided in Inheritance and Hashing. The Trees example was not annotated at all – it was presented only as an outline of activity types and titles. The Moodle examples contained no annotations. Importantly, the examples provided to the PACT and Moodle conditions were identical except for the presence or absence of annotations – the sequence of activities, activity types, and activity contents were consistent across conditions.

8.1.4 Post-Task Survey

After completing the design task, participants were given a survey covering their experience with the design tool (PACT or Moodle) and their interactions with the four examples. The experience portion of the survey covered learning (both about LCI design in general and specifically with respect to patterns) and general opinions of the examples. The examples-oriented portion of the survey was broken into four parts, one for each of the examples. In each of these parts, the participant was asked whether or not they referred to this segment at all while doing their design task. If they did refer to the segment, they were asked to provide a brief summary of the pedagogical approaches used by the author of the course segment. They were then asked a series of subjective questions about their perceptions of the example segment and the ways that they used the example to inform their own design work.

8.1.5 Summary Interview

The final portion of the user study was a brief semi-structured interview. During this interview, I asked each participant for more information about their experience with the design tool, how they incorporated examples, and how confident they were in the strength of their course design.

8.2 Results & Discussion

The background surveys confirmed that there were no significant differences between the treatment and control groups with respect to prior teaching experience or attitudes towards teaching. Participants had taught or served as a teaching assistant an average of 3.6 semesters. Each participant had significant programming experience, with no significant differences in prior experience with C between the treatment and control groups.

8.2.1 Learning from Example Designs

PACT was designed to help users learn about best practices in LCI. In PACT's example-driven approach to learning, the fundamental building blocks are an understanding of each course design, what makes it a good design, and its creator's design intentions. I evaluated how PACT and Moodle impact this understanding in several ways.

8.2.1.1 Expert Scores of Participant Summaries

My first evaluation of this understanding is on a per-example basis using the participant's summary of the pedagogical approaches used by the author of the example. Each of these responses was collaboratively evaluated by a group of three expert LCI curriculum developers and assigned a score from 0-5, measuring its accuracy and completeness. Due to time constraints, most participants did not examine all four sample course segments. For this analysis, I omitted participants who did not look at the relevant course segment (per self-report).

Course	Annotation Level in PACT	PACT Median	Moodle Median	U $(n1,n2)$	p
Inheritance	High	**1.5**	0.0	85.5 (10,10)	< 0.05
Hashing	High	**2.5**	1.0	74.5 (10,9)	< 0.05
Recursion	Low	**2.0**	1.0	73.0 (9,11)	< 0.1
Trees	None	0.5	**1.0**	55 (8,9)	> 0.16

Table 8.2: Scores for participant-generated example course summaries for each conditions and example course pairing. Scores were assigned by expert evaluators on a 0-5 point scale. Significance levels were calculated using a two-tailed Mann-Whitney U test.

The PACT approach has a clear and significant impact on how well users understand the pedagogy of examples. Across the three curriculum segments that were annotated in PACT, the median summary scores for the PACT and Moodle conditions were 2.0 and 0.5, respectively. The distributions of these two groups differ significantly with high confidence (Mann-Whitney $U = 744.5$, $n_{pact} = n_{Moodle} = 30$, $p < 0.01$). This is in contrast to self-reported scores, where both PACT and Moodle participants indicated that they easily understood the segments, providing a median value of 4 on a 5-point Likert scale for each condition with no significant difference between the two distributions (Mann-Whitney $U = 548.5$, $n_{PACT} = 31$, $n_{Moodle} = 30$, $p > 0.22$). This matches my field observation that novices often believe they thoroughly understand example courses even if objectively they do not.

Results for each individual example course segment varied with the level of annotation provided in the PACT condition. The outcome for each example is summarized in Table 8.2, which includes the median scores for the PACT and Moodle conditions along with significance levels. Score distributions for the highly annotated Inheritance and Hashing examples differed significantly between the two conditions ($p < 0.02$). The distributions were marginally significantly different for the lightly annotated Recursion example ($p < 0.08$). For the Trees example, which had no annotations in the PACT condition, there was no statistically significant difference in summary scores between the two conditions ($p > 0.16$) and the Moodle condition participants had the higher median score. This is strong evidence to support my claim that the PACT display of pattern annotations enables a richer understanding of example course designs than examining them as they

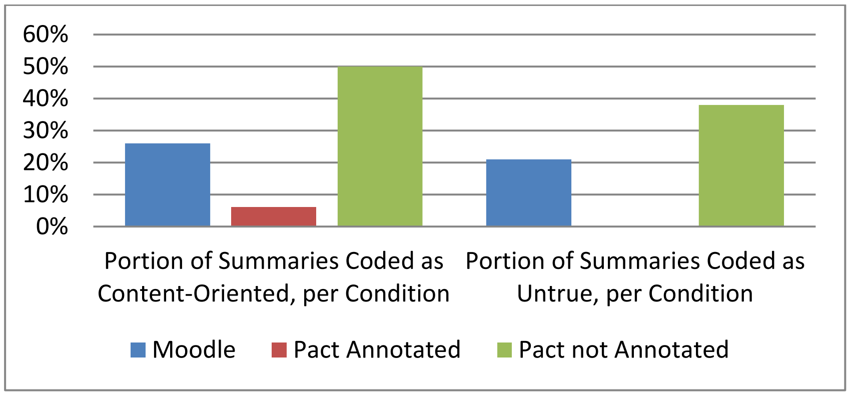


Figure 8.1: Recurring Trends in Participant Summaries

are traditionally shared amongst instructors.

8.2.1.2 Recurring Trends in Participant Summaries

Two additional trends emerge when the individual summaries are examined (summarized in Figure 8.1). First, Moodle participants were much more likely than PACT participants to focus exclusively on the *types* of content being employed in a course segment at the exclusion of discussing the *purpose* of the content. Considering only the three annotated examples, 8 of 30 (27%) Moodle summaries focused exclusively on content types compared to 2 of 30 (6%) PACT summaries. An example Moodle participant summary of the Hashing example exemplifies this issue: "Mostly giving info. The equivalent of lecture with some short quizzes throughout and lots of use of forums among the students." A PACT participant summary of the same example shows a more nuanced understanding of *why* those activities exist: "This author works by following up brief periods of text with lots of examples. The students in this class would spend lots of time testing out new ideas and relatively little time reading. Students 'self-test' new ideas by trying out new things they have just read about in groups."

The second clear trend in the summary responses was the prevalence of objectively counter-factual summaries provided by Moodle participants. Again considering only the three annotated examples, Moodle participants provided clearly untrue summaries in 7 of 30 (23%) attempts while PACT participants provided no obviously untrue summaries. For this analysis, we considered a summary to be untrue only if it is wholly inaccurate about a fundamental issue related to the pedagogy of the example – we did not look for minor inaccuracies. A Moodle participant's summary of the Inheritance example demonstrates the issue: "Almost exclusively lecture-type material. Very little feedback or activity-based learning." This claim is demonstrably false, as the example contains considerable active content with a good mix of feedback and exercises.

These trends did not hold for the Trees example, which was not annotated in the PACT condition. For that example, PACT participants were actually more likely to focus on content types or to make erroneous claims than were Moodle participants. 4 of the 8 (50%) PACT summaries focused on content types compared to 2 of the 9 (22%) Moodle summaries. 3 of the 8 (38%) PACT summaries were wholly inaccurate compared to 1 of the 9 (11%) Moodle summaries. I suspect that these effects come from PACT participants becoming reliant on the annotation system for comprehending the examples. A lack of structural annotations on the Trees example may have caused participants to believe there truly was no purpose for the individual elements, causing them to fall back to content types when summarizing. Similarly, when certain tags (e.g. Active Content and Exercise) were not evident in the display the participants may have believed that those elements were not present in the course, leading to untrue summaries like: "This segment felt like a lecture: lots of explaining material and little in the way of exercises or self-assessment."

Prompt	PACT Median	Moodle Median	*U* *(n1,n2)*	*p*
The tool helped me learn to be a better curriculum designer.	**4.0**	3.0	124.5 (13,13)	< 0.05
The example course segments generally helped me complete my task.	**4.0**	3.0	118 (13,13)	< 0.1
I spent a lot of time examining the example course segments.	**3.0**	2.0	126.5 (13,13)	< 0.05
I now know more about pedagogical design patterns.	**4.0**	3.0	126 (13,13)	< 0.05
I examined this curriculum segment while completing the design task.	**4.0**	3.0	977 (39,39)	< 0.05
It was easy to determine why most of the steps of this example were included in the design and what their intended purposes were.	**4.0**	4.0	591.5 (31,30)	< 0.1
I drew inspiration from this example for my own design.	**4.0**	3.0	581 (31,30)	< 0.1

Table 8.3: Participant responses on the post-task survey that are relevant to learning from example designs. The top four rows are from the general survey. The bottom three are aggregations of the example-specific surveys for the three courses that were annotated in PACT (Hashing, Inheritance, and Recursion). Significance from two-tailed Mann-Whitney U test.

8.2.1.3 Self-Reported Thoughts on Learning from Examples

A final window into how participants learned from examples while designing curricula with PACT and Moodle comes from self-reported survey responses and interview comments. Table 8.3 summarizes responses to relevant questions on the post-task survey (rows 1-4) and aggregated values from the per-example surveys of the three annotated examples (rows 5 – 7). PACT participants reported that the design tool helped them learn to be a better curriculum designer significantly more frequently than did Moodle participants. PACT participants generally felt more engaged with the examples and believed more strongly that the examples helped them to complete their design task. Unsurprisingly, PACT participants stated that they knew more about pedagogical design patterns after completing the task than they did beforehand significantly more than did Moodle participants.

Further, PACT participants rated the annotated examples as being useful significantly more than the Moodle participants did. This effect was seen on questions about how much the participant examined the curriculum example, how easy it was to rationalize about the contents of the example, and how much the participant drew inspiration from each example while designing their own course segment. Similar differences are seen when comparing PACT participant ratings of the utility of the annotated examples versus the un-annotated Trees example.

All of my findings with respect to learning from and comprehending examples with PACT and Moodle are supported by comments made by participants during summary interviews. Representative comments from PACT participants include:

- "Tags and notes were **necessary** to understand the content. I don't see how I could have done this without them."

- "[The un-annotated Trees example] didn't show the thought process. It was like just reading the course… it didn't explain why we wanted to do things a certain way."
- "With the tags and annotations it's easier to see things. When I scan [the course], I can quickly tell everything's purpose."

Comments from Moodle participants include:

- "The Moodle content types are like file types on a computer. Looking at them tells me nothing about what the content does or what its purpose is."
- "Really, I had no idea what was going on in any of the examples. I made my course based on my gut feeling of how a lab-centric course should look."

8.2.2 Persuasiveness of Examples

The PACT interface and annotations did not just impact the participants' comprehension of example course segments; they also affected perceptions of the examples and their creators.

8.2.2.1 Perceived Quality of Examples

PACT participants were significantly more likely than Moodle participants to believe that the annotated examples were of high quality, as shown by their responses on a 5-point Likert scale to the question "I believe that this curriculum segment exemplified best pedagogical practices." PACT and Moodle participants answered with median scores 4.0 and 3.0, respectively, and the difference between the distributions was highly significant (Mann-Whitney $U = 752$, $n_{Moodle} = n_{PACT} = 31$, $p < 0.001$).

Interestingly, the same effect is present when comparing PACT participants' reactions to each of the annotated examples against the un-annotated Trees example – they were significantly more trusting that each annotated example exemplified best practices. For instance, PACT participants scored the Hashing and Trees examples with median scores of 4.0 and 3.0, respectively, with significant differences between the distributions (Mann-Whitney $U = 79$, $n_{Hashing} = 11$, $n_{Trees} = 8$, $p < 0.01$). No similar effect was seen in the Moodle condition. This suggests that exposure to pattern-annotated courses makes un-annotated courses seem lower quality by comparison. At very least, participants became accustomed to the annotations in the other examples and did not invest the time needed to thoroughly explore the un-annotated course. This particular difference was claimed in an interview by a PACT participant: "It's just obvious that the Trees example is not as good as the Hashing example. It's just visually clear that the Hashing example is better… more interactive." This participant was surprised to learn that the two examples were written by the same author, come from the same portion of a single course, and employ nearly identical pedagogies.

8.2.2.2 Perceived Competence of Example Authors

PACT participants were also significantly more likely than Moodle participants to believe that the authors of the annotated examples were competent curriculum designers and instructors. This statistically significant effect can be seen in responses to the following questions, each answered on a 5-point Likert scale:

- I would want to take a class from the author of this example.
    - $Median_{PACT}$ **4.0**, $Median_{Moodle} = 3.0$, Mann-Whitney $U = 665$, $n_{Moodle} = 31$, $n_{PACT} = 30$, $p < 0.01$
- I would happily TA for the author of this example.
    - $Median_{PACT}$ **4.0**, $Median_{Moodle} = 3.0$, Mann-Whitney $U = 615$, $n_{Moodle} = 31$, $n_{PACT} = 30$, $p < 0.05$
- If I were developing a new course, I would gladly accept advice from the author of this example.
    - $Median_{PACT}$ **4.0**, $Median_{Moodle}$ *4.0,* Mann-Whitney $U = 615$, $n_{Moodle} = 31$, $n_{PACT} = 30$, $p < 0.01$

Again, this same effect can be seen within the PACT condition when comparing the annotated examples with the un-annotated Trees example. For instance, on the "accept advice" question the responses for the Hashing and Trees examples had median scores of 4.0 and 3.0, respectively, with highly significant differences between the distributions (Mann-Whitney $U = 84$, $n_{Hashing} = 11$, $n_{Trees} = 8$, $p < 0.001$).

8.2.3 Quality of Developed Curricula

A final mechanism for measuring the effectiveness of PACT and pattern annotations is to evaluate the actual course segments created by participants during the design task portion of the study, along with their self-reported thoughts on the design process and outcome.

8.2.3.1 Expert Scores of Developed Course Segments

The course segments created by participants during the study were converted into Moodle-only versions, stripping away any annotations created by PACT participants and removing any evidence that would point to which design tool was used in the creation of the segment. One course segment designed by a Moodle participant was excluded from this stage of the analysis due to the participant's having creating a course segment that was not evaluable by the experts (this participant's data for other questions remained valid). Thus, a collection of 13 PACT-created designs and 12 Moodle-created designs remained for evaluation.

These 25 designs were given to two LCI experts to be collaboratively scored on eight different criteria, each of which is a key facet of developing good lab-centric curricula. The experts provided scores from 0-5 for each criteria, leading to a total possible score range of 0 – 40 for each design. Based on these totals**,** the designs created by PACT participants were generally of higher quality than those created by Moodle participants. Participants in the PACT and Moodle conditions had median scores of 31.0 and 23.5, respectively, with marginally significant differences between the distributions (Mann-Whitney $U = 110$, $n_{PACT} = 13$, $n_{Moodle} = 12$, $p < 0.1$).

Among the eight criteria used for scoring the designs, four showed significant effects between conditions, with PACT participants scoring higher in all four. They were:

- *C1: Student is Sufficiently Active*: Is the student actively involved in constructing knowledge, as opposed to just reading or watching videos? Ideally, about 50% or more of the student's time should be spent doing something active.
- *C2: Appropriate Use of Embedded Assessments*: Does the segment feature sufficient embedded assessments? Are opportunities presented to help students quickly realize whether they do

or do not understand material? Are assessments in-line with the other material (formative), rather than bunched together at the end of the day (mastery)?

- *C3: Proper Diversity of Programming Activities:* Are a variety of programming activities employed? Is it all coding, or is there some debugging, reading of code, tracing, etc?
- *C4: Proper Cohesiveness and Difficulty Gradient:* Does the course feel right in terms of the difficulty of learning the material? Does the lab day progress cohesively from easy material to difficult material?

Criteria	PACT Median	Moodle Median	U	p
C1	**4.0**	2.0	113.5	< 0.06
C2	**4.0**	2.0	109.5	< 0.1
C3	**2.0**	1.0	114.5	< 0.05
C4	**3.75**	3.13	117.5	< 0.05

Table 8.4: Expert-assigned scores for four criteria used to evaluate the design of lab-centric courses (described in the text). Significance levels from two-tailed Mann-Whitney U test, $n_{PACT} = 13$, $n_{Moodle} = 12$.

While statistically significant results were not obtained for the other four criteria, PACT participants generally had higher scores (as measured by median and mean scores) on those as well. Ultimately, a lack of significance in some of the categories is not surprising and it seems likely that a study with more participants would produce statistically significant results for the remaining criteria. The curriculum design task was very complicated and the measurable outcome (the designed course) is a very noisy signal – that statistically significant results for four of the criteria emerged from this noisy signal is a pleasant surprise.

Results for these four significant criteria are summarized in Table 8.4. These results show that the PACT interface has realized many of the potential benefits of applying pedagogical design patterns to the curriculum design process. The positive result on *C1* is likely due to a combination of trend patterns and the high visibility of the Active Learning tag in example designs. Both interface features provide constant reminders that Active Learning is highly desirable and the dominant feature of a well-designed lab-centric course. The result on *C2* is likely due to the prevalence of the Self-Test multi-step pattern throughout the example courses. By copying and cloning from existing courses, participants could successfully maintain this pedagogical technique while changing the content and activity details to match the new context.

The *C3* result is attributable to the high visibility of exercise-related patterns and tags within the PACT display. The various exercise types each have their own tags and are incorporated into pedagogical patterns that make optimal use of their unique features. These features (along with trend pattern annotations) help PACT users maintain a proper diversity of activities throughout a design. Finally, *C4* brings all of these concepts together. A user that has a better understanding of proper lab-centric design, is guided by carefully crafted annotations, and is constantly visually reminded of what elements go into a good design is much more likely to create a cohesive lab-centric experience for their students.

8.2.3.2 Self-Reported Thoughts on Curriculum Development

Participants in each condition generally believed that they had done equally well on the curriculum design task. PACT and Moodle participants responded to the 5-point Likert scale question "I created an excellent curriculum segment" with median scores of 3.0 for each condition. For the question "The design task was easy to complete" the median scores were also 3.0 for each question. (There were no meaningful differences between the score distributions per condition for either question). These results match with my field observation that novice curriculum designers generally feel that they are doing well, regardless of what an objective measure would say.

Participant comments give some insight into what features of the tools were and were not helpful. PACT participants generally felt that PACT was pushing them in the direction of making a specific type of course: "The tags funnel me into one way of teaching the course" and "It gives you no choice but to do a better job." Some users found this helpful while others found it somewhat constraining. In both cases, it appears that the users ultimately benefitted from these features. Moodle users, by contrast, felt that they could create whatever type of course they wanted to: "I felt free to build the course the way I saw it in my head," but suffered somewhat as a result: "My course is all over the place. I have no idea if what I did was right."

PACT users appreciated the organizational utility of structural annotations and tags: "Tags were reminders of what needed to be occurring… why each portion exists… what the big picture was" and "if I wanted an outline like [the un-annotated] Trees example I could just scribble it down on a piece of paper. It doesn't have the benefits that come from the tool – helping to organize my thoughts and teaching strategy." As comparison, Moodle users found the process of building their course segment to be somewhat too unstructured: "I was lost the entire time" and "I was never sure what to write next or how to connect it to what I had just written."

PACT users found that tags and pattern annotations (especially trend patterns) helped them maintain a good diversity of activity types and purposes: "Tags force me to think about how to make this segment different from the previous" and "[trend patterns] are a good reminder of the different types of content to include. It showed the diversity that was desirable in the design." As is shown by the experts' scores, Moodle users had trouble maintaining high levels of desirable content types (such as Active Learning) and proper content diversity. One frustrated Moodle user summed up their approach thusly: "I just went with the activity type that seemed the most familiar to me (HTML pages)."

Chapter 9

Future Work

This dissertation has presented the VDA model for promoting design learning via design tool-mediated interactions with annotated design artifacts, a mechanism for converting best practices and design patterns into annotations, and PACT, an instantiation of the VDA framework in a fully implemented and evaluated overlay curriculum design tool. In this chapter, I describe potential future advances beyond this thesis for each of these elements of my work, building up from the concrete PACT tool and out to the general VDA framework.

9.1 The Future of the PACT Overlay Design Tool

The primary avenue of future work for the PACT tool is real use. PACT has been designed and developed to a point where it can (and should) be used by university instructors as they create new courses and build their own understanding of the principles of learner-centered design. In this section, I describe additional aspects and applicability of PACT that make it a practical design tool ready for incorporation into professional design practice. I focus on the community features of PACT that complete its instantiation of the VDA framework and the ways in which PACT could be used by instructors in various instructional contexts.

9.1.1 Community Features

In the user study described in Chapter 8, I used example annotated courses that I produced to serve as a surrogate for community-produced annotated artifacts. While establishing, sustaining, and evaluating a real community of practice around annotated course development is beyond the scope of this dissertation, the design principles required to support this community are included within the Virtual Design Apprenticeship framework presented in Chapter 2. Further, I have designed and implemented features in PACT that realize these principles. In this section, I present these features in brief to demonstrate how an overlay design tool can integrate support for a community.

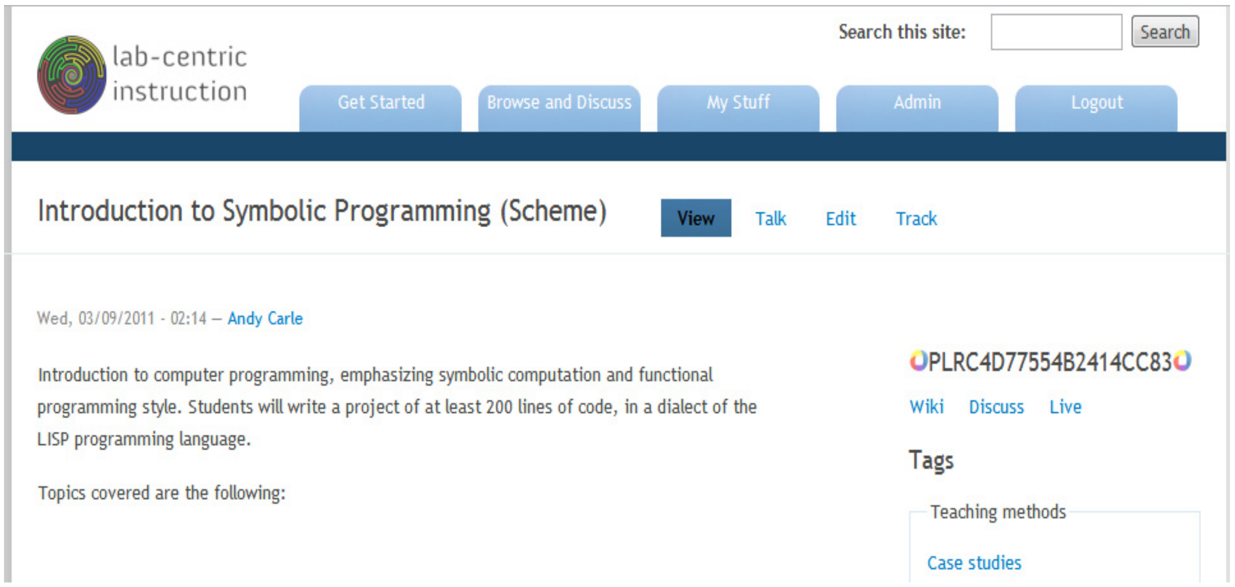
9.1.1.1 Annotated Course Repository & Search Tools

The fundamental community-oriented design principle in VDA is that design tools must **facilitate sharing annotated design artifacts and finding relevant designs.** Towards this end, PACT2 features a public repository of annotated courses which allows the user to browse the collection of contributed PACT XML documents. Users can peruse the repository from within the authoring tool or via the PACT community website (discussed below). The collection can also be explored via a faceted search (Hearst, 2006) with which the user can filter the course repository to find relevant examples. Examples of the facets provided for filtering include design pattern annotations, course-level keywords, and individual task-oriented tags. Imagine a novice user who is learning about the Self-Test pattern, having been inspired by an example course segment to try it in his own design. He could use the search tool to quickly find other courses that include the self-test pattern annotation or look at specific learning activities tagged with the Self-Test tag. Ultimately, this feature is intended to make expert-contributed annotated courses more useful to the community as a whole and to provide increased visibility for helpful examples.

9.1.1.2 Community Website with Two-Way References

The PACT project includes a community website that complements the authoring tool and acts as a collaboration and community communication layer. It provides a more accessible portal into the PACT repository than the PACT authoring tool itself and can help potential users ease into the PACT project. The website is implemented as a minimally-modified instance of the Drupal open-source content management framework. It includes standard community features like forums, wikis, and messaging. I have augmented these features to include special content types and tools that deal with pattern annotated courses and course contents, such as the previously described faceted search engine.

The most interesting of these augmentations is a two-way linking system built to help users move quickly between Drupal community features and the PACT authoring tool. Users can generate links from any view of any artifact with PACT; these links are implemented as a unique text key that is placed onto the user's clipboard. The user can then include these links into any content on Drupal, such as a wiki page, a forum post, or as a comment on a course design. The link will then be automatically styled to stand out from surrounding text, as shown in Figure 9.1. Other users can follow the link back into the PACT editor and see the exact view from which it was generated. This allows users to ground community discussions in real design artifacts, helping to keep the discourse concrete and avoid the sorts of abstract discussions that novices have trouble following. Thus, the PACT community website, and this feature in particular, are instantiations of the VDA design principle to **structure community discussions around the annotated design artifact.**



Introduction to Symbolic Programming (Scheme)

Wed, 03/09/2011 - 02:14 — Andy Carle

Introduction to computer programming, emphasizing symbolic computation and functional programming style. Students will write a project of at least 200 lines of code, in a dialect of the LISP programming language.

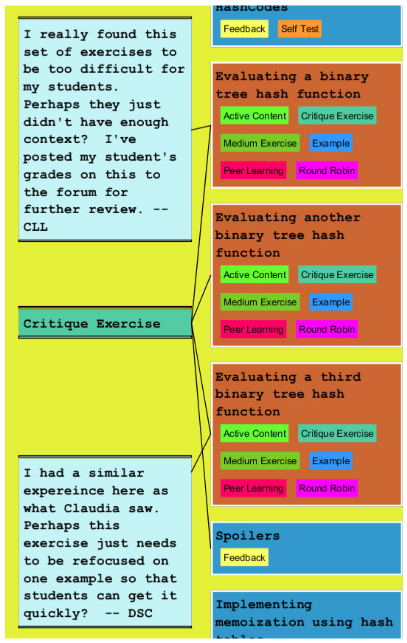
Topics covered are the following:

PLRC4D77554B2414CC83

Figure 9.1: The PACT webpage corresponding to a specific annotated course. The PLR link on the right (enclosed with PACT logos) can be used in the PACT editor to open the relevant course. Similar references can be generated for specific views or pattern annotations. These references can be embedded in any pages, comments, and forum posts on the site to facilitate discussion.

9.1.1.3 Student Data Annotations, Design Feedback & Design Use Agreements

The final VDA design principle to be covered by a PACT feature is to **encourage learners to provide performance feedback to experts.** This principle is realized in PACT2 by employing special structured annotations. Rather than being based on pedagogical patterns, these annotations are crafted by expert instructors to help gather data back from other instructors who employ the expert's designs. For instance, an expert might include an annotation on a learning sequence that includes a spot for other users to include data on how well received that sequence was by students or to include student responses to a particular quiz or brainstorm activity. While not yet implemented, more complex data gathering instruments could automatically pull this type of data out of the associated LMS, allowing expert designers access to student responses without any intervention on the part of the instructor using the design. This would be relatively easy to implement via a PACT2 plugin. Additionally, all users can provide simple, unstructured design feedback via the PACT community website and with freeform annotations in the authoring tool, as shown in Figure 9.2.


I really found this set of exercises to be too difficult for my students. Perhaps they just didn't have enough context? I've posted my student's grades on this to the forum for further review. -- CLL

I had a similar expereince here as what Claudia saw. Perhaps this exercise just needs to be refocused on one example so that students can get it quickly? -- DSC

Figure 9.2: Two users have provided design feedback to the expert who created this course. The expert will receive these comments and know precisely which parts of the course did or did not work well.

A planned, but not yet realized, extension of these concepts is to incorporate an optional contract-structured model for sharing expert content within PACT. In this system, users would have to promise some degree of feedback to experts before being allowed to use their content. A user's fulfillment of these obligations would contribute to a score within the PACT website, similar to how reputation point systems (Resnick, Kuwabara, Zeckhauser, & Friedman, 2000) work on popular community-driven question-and-answer websites like stackoverflow.com. Regardless of the ultimate implementation, the goal of these features is simply to connect experts who have good design ideas with novices in need of assistance designing a new learner-centered course. This completes the cycle, making PACT as a whole useful to novices, experts, and everyone in between.

9.1.2 Supporting Instructional Modalities Other Than Lab-Centric Instruction

This dissertation has focused on PACT as a design tool to be used in conjunction with the Lab-Centric Instruction teaching modality. While LCI was chosen as the platform for this research for good reasons (as discussed in Section 3.3), nothing about the VDA framework or the design of the PACT tool is tied to LCI in any fundamental way. VDA applies to any design artifact that can be represented visually and edited with design tools. PACT can, in principle, work as an overlay design tool for any learning platform. For instance, extending PACT to edit the contents of a Massively Open Online Course system would be a simple matter of writing the publishing plugins necessary to interact with that system's editing APIs. Integrating MOOCs would, indeed, be an excellent next step for PACT, as most of the pattern annotations already in the PACT Annotation Library would apply just as well to MOOC content as to LCI content.

In a more radical adaptation, PACT could even be used as a design overlay for lecture-based courses. The learning activities in the PACT visualization would map neatly to the individual slides in a presentation. Such activities need not be just the instructor preaching to a classroom, as in traditional lectures – simple in-class activities like peer instruction, group collaborations, and even small exercises all can potentially be instantiated within the lecture context. Many (though not all) of the pattern annotations already incorporated into PACT would apply to this type of hybrid "learner-centered lecture" environment. By facilitating this type of learning modality, PACT could help instructors who are unwilling to convert entirely to LCI incorporate better pedagogy into their instructional design. This small step could help the instructor gradually build an understanding of learner-centered teaching methods in a safe and familiar context.

9.2 Expanding the PACT Annotation Schema & Library

Future work on the PACT Annotation Schema centers around making the mechanism more robust, allowing for the generation of more accurate and complete annotations based on pedagogical patterns. One example of this type of improvement, focusing on annotation constraints, is described in this section. The future of the PACT Annotation Library lies in expansion – first by incorporating new pedagogical patterns as they are discovered and formalized; second by incorporating patterns from sources other than the Pedagogical Patterns Project. This future growth is described in detail within this section.

9.2.1 Additional Annotation Constraints

In Section 6.2, I described three annotation constraints that can be included in the definition of a PAS pattern: content type constraints, frequency constrains, and distance (or timing) constraints. While these examples have proven very useful within PAS and the PACT tool, they only begin to cover the potential for building annotation constraints into the PAS system. In this section, I describe two potential extensions of PAS that would incorporate additional aspects of a pattern's definition. Many other types of constraint would be useful and appropriate, but are beyond the scope of this dissertation.

One prominent example of a useful constraint that has not yet been implemented in PAS is a constraint on *when* (in absolute, rather than relative) terms a pattern should be instantiated in a course. A pattern that would obviously benefit from this constraint mechanism is Early Bird (Bergin, Eckstein, Manns, & Wallingford, 2001), summarized in PAS as: "Organize the course such that the most important topics are taught first. Teach the big ideas first and revisit them often." Early Bird implies one content tag: Big Idea. Currently, Early Bird appears in the PACT Annotation Library as a trend annotation with a special note that it should only be applied to containers that occur early in the course. A more formal mechanism for capturing this quality would be useful to both experts and novices working with this pattern.

As a second example, a system for representing context and forces within pattern annotations would be extremely useful, if rather complicated. Most pedagogical patterns include some context describing the situations in which using that pattern is appropriate. For instance, the War Game pattern (Bergin, Eckstein, & Sharp, 2002b) suggests that its usage is appropriate when "you wish to introduce learners to realistic development scenarios." This aspect of the pattern is not captured by the current PAS mechanism, nor is it available to be presented within the PACT interface – either experts must explicitly note this rationale for including the pattern within a note or novices must derive this intent from reading the full pattern text. Extending PAS to include such information (and developing reasonable presentation mechanisms) would make pattern-annotated courses more useful for both novices and experts.

9.2.2 Generating Annotations from New Pedagogical Patterns

The body of patterns provided by the Pedagogical Patterns Project is not – and never could be – a complete description of the space of learner-centered instructional design. There are surely essentially infinitely more excellent teaching practices in the world – each one a potential pedagogical pattern waiting to be discovered and formalized. Some of these future patterns would cover the proper use of new classroom technologies, new teaching platforms (such as Massively Open Online Courseware), or simply be new codifications of existing teaching practices. PACT was designed to facilitate the discovery of these potential patterns by encouraging and structuring reflection on the part of expert designers (as discussed in Chapter 5).

When these new patterns are produced by the education community, they can be easily converted into annotations using PAS and incorporated into the PACT Annotation Library. This process is not dependent upon future patterns exactly matching the format employed by the Pedagogical Patterns Project – the PAS formalization is intended to be generally applicable to any collection of pedagogical design patterns with any type of structure. For instance, Linn and Eylon's Scaffolded Knowledge Integration-based instructional design patterns (M. C. Linn & Eylon, 2011), which bear little resemblance to the patterns of the Pedagogical Patterns Project, can all be decomposed into tags and multi-step patterns. Each knowledge integration process specified by a pattern can be turned into a unique tag that can be applied to content. The sequence and specific details of the pattern would lead to multi-step pattern annotations and associated constraints.

9.3 Extending Virtual Design Apprenticeship

The most important future work on the Virtual Design Apprenticeship framework is to build additional design tools that instantiate the VDA principles. While the focus of this dissertation has been on instructional design, design tools can leverage annotated design artifacts to promote design learning of any type and in any domain. Potential future projects could include integrating learning with overlay annotations in a CAD tool for mechanical and industrial engineers, a visual layout tool for user experience and interaction designers, an integrated development environment for software engineers, or even a text editing document for writing academic papers or dissertations. Each of these design areas comes with best practices and principles that guide good designs and that must be learned by newcomers to the field. The VDA model is an excellent fit for them all.

Further, there is room for improvement within the VDA model and its principles. As our collective understanding of how people learn and how designers work continues to evolve, so too must the VDA framework. In the remainder of this section, I describe one such extension of VDA intended to incorporate a more long-term view of the design process and how designs evolve over time.

9.3.1 Incorporating Empirical Performance Data into the VDA Model

Virtual Design Apprenticeship, as presented in Chapter 2, incorporates a rich ecology of design feedback in the form of design critiques from expert and peers. This feedback model is limited, however, in that it occurs entirely *a priori* – that is, the VDA model only considers iterative improvements to designs before they are put into production and used in the real world. A useful (arguably necessary) extension to VDA would be to incorporate *a posteriori* feedback, framing additional iterative design cycles within the context of the successes or failures the design achieved in real use. For instance, a design tool for mechanical engineers could incorporate information on failure rates or actual production costs of each component of a design. This data could be presented as a new type of annotation, based not on design principles but on real-world performance data. This notion can be formalized as a 10th design principle for VDA design tools: **inform iterative design improvements with empirical performance data.**

As an example, consider a potential implementation of this principle within the PACT tool. Each semester, an instructor gathers extensive data about their students and their course. This data may come in the form of measures of student performance (e.g. grades), qualitative feedback (e.g. survey responses), or general notions of what parts of the course worked and what parts did not. PACT could provide affordances that allow the user to record this information as an additional annotation layer on top of the course design. In fact, PACT already facilitates this interaction in a primitive way via free-form note annotations. A more sophisticated implementation would use a PACT plugin to automatically pull relevant data out of the associated learning management system. This data could be summarized and presented to the user as tags applied directly to content, for instance a "Students Struggled" tag on content where student outcomes were poor or a "Too Long" tag on content that took too much of the day's lab time. This extra information would help focus the instructor's efforts when improving the course for subsequent offerings.

Chapter 10

Conclusion

Design fields are constantly changing – shifting and evolving to incorporate new practices and address new contexts. Professional designers *must* keep pace with these changes in order to remain relevant, but it is difficult to fit formal education around a busy work schedule. In this dissertation, I have argued for a learning model and related software tools designed to enable this lifelong design learning. Called Virtual Design Apprenticeship, this model envisions novice designers as active learners working to construct a robust understanding of a design space and its best practices. The crux of my approach is an integration of practical design tools and learning tools, a combination that makes the resulting system more useful than either could be on its own. The key feature of VDA is a systematic focus on annotated design artifacts – concrete representations of real designs that have design rational overlaid on top of them. This combination exposes expert design thinking and makes it available for consumption by novices.

I have presented university-level instructional design as an ideal testing ground for the VDA framework. This domain is significant because the potential learning benefits of moving from lecture-based courses to learner-centered instructional modalities are substantial and worth pursuing. The barriers to entry in learner-centered design are equally significant, with most university faculty unable to reasonably commit the required time and effort needed to learn these new instructional design paradigms. Lowering these barriers to entry is crucial to the future of tertiary education.

I described the contextual design process that propelled my work over six years and summarized some of the observations from that study. That study influenced both the formulation of VDA and the initial design of PACT, a curriculum design tool that leverages the learning principle of making thinking visible to assist learners as they transition from concrete to abstract reasoning about curriculum design. I detailed the applications of this tool in the curriculum design process and demonstrated that it helped expert curriculum designers reflect and iterate over their curricula.

I detailed the strengths and weaknesses of the PACT system, with a particular focus on how *ad hoc* pattern annotations are a deficient tool for building pattern annotated courses. To solve this problem, I presented the PACT Annotation Schema, a formal mechanism for translating the freeform text of a pattern description into a concrete annotation that can be applied directly to content. I used this schema to generate a library of annotations (based on the publically available patterns of the Pedagogical Patterns Project) that is used within the PACT editor. Further, I offered a critique of the different types of pattern offered in the Pedagogical Patterns Project and their utility in the curriculum design process.

I walked the reader through a redesigned version of the PACT system, showing how it has evolved over time to embody the design principles of the VDA framework. I demonstrated, through a formal user study, PACT's significant impact on how novice designers learn from expert-generated examples, how they perceive the credibility of those examples, and the quality of curriculum designs those novices can produce. Finally, I detailed future work beyond this dissertation, discussing how VDA can and should be applied to design fields beyond instructional design, how pattern annotations can be generated from a wider body of best practices, and the final pieces of the PACT design: features intended to support a community of practice of experts and novices working with pattern-annotated design artifacts. These final features make PACT a complete realization of the VDA framework, delivering on all of its potential to promote learning of instructional design via overlay design tools.

Bibliography

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

Arias, E., Eden, H., Fischer, G., Gorman, A., & Scharff, E. (2000). Transcending the individual human mind—creating shared understanding through collaborative design. *ACM Transactions on Computer-Human Interaction (TOCHI), 7*(1), 113.

Barab, S. A., MaKinster, J. G., & Scheckler, R. (2003). Designing system dualities: Characterizing a web-supported professional development community. *The Information Society, 19*(3), 237-256.

Baudisch, P., Good, N., Bellotti, V., & Schraedley, P. (2002). Keeping things in context: A comparative evaluation of focus plus context screens, overviews, and zooming. *Conference on Human Factors in Computing Systems: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves,* Minneapolis, Minnesota, USA. *, 20.* (25) pp. 259-266.

Beckman, S. L., & Barry, M. (2007). Innovation as a learning process: Embedding design thinking. *California Management Review, 50*(1), 25.

Bederson, B. B., Grosjean, J., & Meyer, J. (2004). Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering, 30*(8), 535-546.

Bergin, J. (2000). Fourteen pedagogical patterns. *5th European Conference on Pattern Languages of Programs, Irsee, Germany, 11.* (June) pp. 2006.

Bergin, J., Eckstein, J., Manns, M. L., Sharp, H., & Sipos, M. (2003). Teaching from different perspectives. *Eighth European Conference on Pattern Languages of Programs.*

Bergin, J., Eckstein, J., Manns, M. L., & Wallingford, E. (2001). Patterns for gaining different perspectives. *PLoP, 2001.* (September)

Bergin, J., Eckstein, J., & Sharp, H. (2002a). Feedback patterns. *EuroPLoP, 2002.*

Bergin, J., Eckstein, J., & Sharp, H. (2002b). Patterns for active learning. *PLoP, 2002.*

Bergin, J., Marquardt, K., Manns, M. L., Eckstein, J., Sharp, H., & Wallingford, E. (2003). Patterns for experiential learning. *EuroPLoP, 2003.*

Booch, G., Rumbaugh, J., & Jacobson, I. (1999). *Unified modeling Language–User's guide*. Reading, MA: Addison-Wesley.

Borgman, C. L. (1999). What are digital libraries? competing visions. *Information Processing and Management, 35*(3), 227-243.

Branham, S. M., Harrison, S., & McCrickard, D. S. (2010). Making design rationale matter: How design rationale has failed and how it can succeed again. *Human Computer Interaction Consortium (HCIC) 2010 Winter Workshop,* Winter Park, CO.

Bransford, J. (2000). In Brown A. L., Cocking R. R. (Eds.), *How people learn: Brain, mind, experience, and school.* Washington, DC: National Academy Press.

Bruner, J. (1977). *The process of education*. New York: Vintage Books.

Clancy, M., Titterton, N., Ryan, C., Slotta, J., & Linn, M. (2003). New roles for students, instructors, and computers in a lab-based introductory programming course. *ACM SIGCSE Bulletin, 35*(1), 132-136.

Collins, A., Brown, J. S., & Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator, 6*(11), 38-46.

Corliss, S. B., Spitulnik, M. W., Higgins, T. E., & Kirkpatrick, D. (2007). Mentored professional development to support successful integration of technology-enhanced science curriculum. *CSCL'07: Proceedings of the 8th Iternational Conference on Computer Supported Collaborative Learning-International Society of the Learning Sciences,* New Brunswick, New Jersey, USA. *, July.* pp. 153-155.

Crouch, C. H., & Mazur, E. (2001). Peer instruction: Ten years of experience and results. *American Journal of Physics, 69*(9), 970-977.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist, 34*(10), 906-911.

Fruchterman, T. M., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software - Practice and Experience, 21*(11), 1129-1164.

Gibbons, J. F., Kincheloe, W. R., & Down, K. S. (1977). Tutored videotape instruction: A new use of electronics media in education. *Science, 195*(3), 1139-1146.

Goel, V. (1995). *Sketches of thought* The MIT Press.

Goodyear, P. (2005). Educational design and networked learning: Patterns, pattern languages and design practice. *Australasian Journal of Educational Technology, 21*(1), 82-101.

Gross, M. D. (1996). The electronic cocktail napkin--a computational environment for working with design diagrams. *Design Studies, 17*(1), 53-69.

Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., et al. (2006). Reflective physical prototyping through integrated design, test, and analysis. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology, October.* pp. 299-308.

Hartmann, B., Morris, M. R., Benko, H., & Wilson, A. D. (2010). Pictionaire: Supporting collaborative design work by integrating physical and digital artifacts. *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, February.* pp. 421-424.

Hartmann, B., MacDougall, D., Brandt, J., & Klemmer, S. R. (2010). What would other programmers do: Suggesting solutions to error messages. *Proceedings of the 28th International Conference on Human Factors in Computing Systems,* Atlanta, Georgia, USA.*, April.* pp. 1019-1028.

Hearst, M. (2006). Design recommendations for hierarchical faceted search interfaces. *ACM SIGIR Workshop on Faceted Search, August.* pp. 1-5.

Hong, J. I., & Landay, J. A. (2007). SATIN: A toolkit for informal ink-based applications. *ACM SIGGRAPH 2007 Courses, August.* pp. 12.

Horner, J., & Atwood, M. (2006). Effective design rationale: Understanding the barriers. *Rationale Management in Software Engineering,* 73-90.

Inglis, A., & Bradley, A. (2012). Using conceptual mapping as a tool in the process of engineering education program design. *Journal of Learning Design, 1*(1), 45-55.

Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., Glaser, D., et al. (2005). Livenotes: A system for cooperative and augmented note-taking in lectures. *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Portland, Oregon, USA.*, April.* pp. 531-540.

Klemmer, S. R., Newman, M. W., Farrell, R., Bilezikjian, M., & Landay, J. A. (2001). The designers' outpost: A tangible interface for collaborative web site information design. *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology , November.* pp. 1-10.

Klemmer, S. R., Thomsen, M., Phelps-Goodman, E., Lee, R., & Landay, J. A. (2002). Where do web sites come from?: Capturing and interacting with design history. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves, April.* pp. 1-8.

Knowles, M. (1984). *The adult learner: A neglected species* (3rd ed.). Houston, TX: Gulf Publishing.

Knowles, M. S., & Bard, R. (1984). *Andragogy in action: Applying modern principles of adult learning*. San Francisco, CA: Jossey-Bass.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development.* Prentice Hall.

Kramarski, B., & Mevarech, Z. R. (2003). Enhancing mathematical reasoning in the classroom: The effects of cooperative learning and metacognitive training. *American Educational Research Journal, 40*(1), 281-310.

Landay, J. A., & Myers, B. A. (1995). Interactive sketching for the early stages of user interface design. *CHI '95: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Denver, Colorado, USA.*, May.* pp. 43-50.

Laurillard, D. (1993). *Rethinking university teaching-a framework for the effective use of educational technology*. St. Louis, MO: Routledge.

Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge: Cambridge University Press.

Lilly, S. (1996, Patterns for pedagogy. *Object Magazine, 5*, 96.

Lin, J., & Landay, J. A. (2002). Damask: A tool for early-stage design and prototyping of multi-device user interfaces. *Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. in Proceedings of the 8th International Conference on Distributed Multimedia Systems, September.* pp. 573-580.

Lin, J., Newman, M. W., Hong, J. I., & Landay, J. A. (2000). DENIM: Finding a tighter fit between tools and practice for web site design. *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* The Hague, The Netherlands. *, April.* pp. 510-517.

Linn, M. C., & Eylon, B. S. (2011). *Science learning and instruction: Taking advantage of technology to promote knowledge integration.* Florence, KY: Routledge, Taylor & Francis Group.

Linn, M. C., & Eylon, B. (2006). Science education: Integrating views of learning and instruction. In P. A. Alexander, & P. H. Winne (Eds.), *Handbook of educational psychology* (2nd ed., pp. 511-544). Mahwah, N.J: Erlbaum.

Linn, M. C., & Clancy, M. J. (1992). The case for case studies of programming problems. *Communincations of the ACM, 35*(3), 121-132.

Linn, M. C., Clark, D., & Slotta, J. D. (2003). WISE design for knowledge integration. *Science Education, 87*(4), 517-538.

MacLean, A., Young, R. M., & Moran, T. P. (1989). Design rationale: The argument behind the artifact. *ACM SIGCHI Bulletin, 20*(SI), 247-252.

Milner-Bolotin, M. (2004). Tips for using a peer response system in a large introductory physics class. *The Physics Teacher, 42*(4), 253-254.

Nov, O., Naaman, M., & Ye, C. (2008). What drives content tagging. *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08; April.* pp. 1097-1100.

Parlante, N. (2007). Nifty assignments. *ACM SIGCSE Bulletin, 39*(1), 497-498.

Piaget, J. (1970). *Genetic epistemology* (E. Duckworth Trans.). New York: American Behavioral Scientist.

Regli, W. C., Hu, X., Atwood, M., & Sun, W. (2000). A survey of design rationale systems: Approaches, representation, capture and retrieval. *Engineering with Computers, 16*(3), 209-235.

Resnick, P., Kuwabara, K., Zeckhauser, R., & Friedman, E. (2000). Reputation systems. *Communications of the ACM, 43*(12), 45-48.

Rittel, H. W., & Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences, 4*(2), 155-169.

Saye, J. W., & Brush, T. (2002). Scaffolding critical reasoning about history and social issues in multimedia-supported learning environments. *Educational Technology Research and Development, 50*(3), 77-96.

Schatz, B., & Chen, H. (1996). Building large-scale digital libraries. *Computer, 29*(5), 22-26.

Schön, D. A. (1987). *Educating the reflective practitioner* Jossey-Bass San Francisco.

Sharp, H., Manns, M. L., & Eckstein, J. (2003). Evolving pedagogical patterns: The work of the pedagogical patterns project. *Computer Science Education, 13*(4), 315-330.

Sharp, H., Manns, M. L., & Eckstein, J. (2000). The pedagogical patterns project (poster session). *OOPSLA '00: Addendum to the 2000 Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (Addendum),* Minneapolis, Minnesota, United States. pp. 139-140.

Staub, F. C., & Stern, E. (1997). Abstract reasoning with mathematical constructs. *International Journal of Educational Research, 27*(1), 63-75.

Titterton, N., & Clancy, M. (2007). Adding some lab time is good, adding more must be better: The benefits and barriers to lab-centric courses. *International Conference on Frontiers in Education: Computer Science & Computer Engineering (FECS 2007),* pp. 363-367.

Titterton, N., Lewis, C., & Clancy, M. (2010). Experiences with lab-centric instruction. *Computer Science Education, 20*(2), 79-102.

Treacy, B., Kleiman, G., & Peterson, K. (2002). Successful online professional development. *Learning & Leading with Technology, 30*(1), 42-47.

Trochim, W. M. (1989). An introduction to concept mapping for planning and evaluation. *Evaluation and Program Planning, 12*(1), 1-16.

Vygotskii, L. S., & Cole, M. (1978). *Mind in society : The development of higher psychological processes / L. S. vygotsky ; edited by michael cole ... [et al.]* Harvard University Press, Cambridge.

Williams, M., & Linn, M. C. (2002). WISE inquiry in fifth grade biology. *Research in Science Education, 32*(4), 415-436.

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17*(2), 89-100.