Pedagogical Patterns – a method to capture best practices in teaching and learning

Eva Magnusson

Abstract— **In this paper we describe a concept that can be used to capture best practices in teaching and learning and for sharing knowledge between educators – The Pedagogical Pattern concept. We present the emergence and work of the Pedagogical Patterns Project. Its goal was initially to document solutions to teaching and learning problems in Computer Science, especially in the field of object-oriented programming. Many of the problems addressed are, however, of a much more general nature and therefore most of the existing pedagogical patterns can be of interest to educators in other fields. Some general patterns are presented as are also some of our own experiences of using patterns in teaching Computer Science.**

I. Introduction

THE English architect Christopher Alexander in 1977 published a book with the title *A Pattern Language* [1], where he uses patterns to document proven solutions to commonly occurring problems in architecture. His definition of a pattern was: *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of a solution to that problem, in such a way that you can use this solution a million times over, without doing it in the same way twice*.

The work of Alexander inspired a group of people involved in software development. In the late 80s they introduced the Design Pattern concept as a way of documenting and communicating solutions to common problems arising in the design of software systems. The background was that a new programming paradigm, the object-oriented technology (OT), had been adopted in industry. Software developers used to other programming paradigms found it very difficult to adapt to OT. Design Patterns proved to be very useful in this context. Collection of Design Patterns were published the most well-known of which is in Gamma et al. [7]. The pattern movement grew rapidly in software industry. Software Design patterns are now also part of the curriculum for most students in Computer Science.

The paradigm shift created a demand for effective teaching and learning of the new technology. This turned out to be a real challenge for Computer science educators. E.g., there was no obvious best order in which to teach the different concepts in object-oriented programming languages. Many of the pedagogical methods used to teach programming languages before did not work. Several different new approaches were discussed in different forums in conjunction with conferences and symposiums dealing with OT. It became clear that there was a need to find effective ways to communicate expertise in this area between university educators. This was the reason for the start of the Pedagogical Patterns Project (PPP) in 1996.

The rest of this paper is organized as follows: In Section II the Pedagogical Patterns Project is presented. Section III describes the Pedagogical Pattern concept and give some example of patterns. In Section IV we show how relationships between patterns can be used to create pattern languages to solve more complex problems. Some of our own experiences of using patterns are summarized in Section V. Finally in Section VI we conclude with a discussion of the pattern concept.

II. The Pedagogical Patterns Project

The motivation for the Pedagogical Patterns Project has been formulated in the following way [8]:

Effectively communicating complex technologies is often a struggle for information technology instructors. They may try various teaching strategies, but this trial and error process can be time-consuming and fraught with error. Advice is often sought from other "expert" instructors, but these individuals are not always readily available. This creates the need to find other ways to facilitate the sharing of teaching techniques between expert and novice teachers.

The goal of the project was thus to create a method to document and share best practices for teaching and learning. The educators involved in the project introduced the concept of a *Pedagogical Pattern* as a means to achieve this. The concept will be presented more in detail in the next section.

Browsing through some of the patterns developed so far, some of the principles on which the project is founded become evident:

- The focus is on students. Patterns discuss teacher activities but it is always clear that the main job of the teacher is to facilitate learning.
- Learning happens best in environments where mentally active processes are supported.
- Students learn differently.

The Pedagogical Patterns Project involves instructors from many different countries and also people from industry. Initially their focus was on collecting reusable techniques and finding a suitable format for pattern mining. Later their focus has shifted to pattern languages (see Section IV) as a more

Eva.Magnusson@cs.lth.se
Department of Computer Science
Lund University

powerful tool for more complex problems. They often gather at workshops in conjunction with conferences dealing with the pedagogy of Computer Science or with Design Patterns for OT.

III. The Pedagogical Pattern Concept

A Pedagogical Pattern deals with pedagogical problems in teaching and learning. The problem space was originally the teaching and learning of OT. It turned out, however, that the problems encountered in this area are common in other disciplines as well. Most of the patterns are therefore quite general and applicable in many different educational contexts.

Pedagogical Patterns focus on practises that have been thoroughly tested and proven useful by several people in different contexts rather than on new ideas They are not something you *invent*, rather they are *discovered*. Sharing these discoveries between educators requires *pattern mining*, which is facilitated by a common documentation format. The project advocates a format where each pattern is divided into sections to make it easy for readers to find the key elements. These include:

- A problem statement
- The problem context and the conflicting forces which create the problem
- A solution, its rationale and consequences
- Concrete examples; evidences that the solution works

Each pattern also has a name, chosen so that it reveals its essence. Examples of pattern names are See Before Hear, One Concept Several Implementations and Built In Failure. Often, the documentation includes references to other related patterns. In Fig 1, you find an example of a pattern called Solution Before Abstraction from [4].

The work of the project is constantly evolving. The evolution of a pattern can roughly be described as having an idea of a good solution to an often recurring teaching problem and present this as a pedagogical pattern draft. In order to have it recognized as a pattern it needs to be discussed in a pattern-writers' workshop, refined, tested and validated. Often the abstraction level of the initial problem formulation is raised during this process in order to make it as general as possible.

A wide variety of patterns have been used and tested by several instructors. They are available at the homepage of the project [8]. From this page there are also links to homepages of the project members where additional suggested patterns can be found.

IV. Pattern Languages

Pedagogical patterns are related to each other in different ways. For example, a pattern can specialize, generalize, parallell, use or complete another pattern. A larger number of patterns can also be interrelated. For example, they can be dealing with solutions to problems arising when teaching a specific subject such as object-oriented programming or they can have a common problem space such as course development or assessment issues. When several patterns in this way collaborate to solve a more complex problem they constitute a *Pattern Language*.

The original goal of the PPP was to form *one* pedagogical pattern language for teaching object technology. Around 2000 it became clear that this domain was too large. Considerable work was done to find and document relationships between the then existing set of patterns. One possible pattern language was identified, The Experimental Learning Pattern [4]. After that three other pattern languages have been developed: Feedback [6], Active Learning [5] and Gaining Different Perspectives [2].

A pattern language documentation always include a quick access table for the included patterns. It lists some problems addressed by the language and the respective patterns that are applicable to solve these problems. Fig 2 shows part of this table for The Experimental Learning Pattern.

In [9] and [10] it is emphasized that the concept of a pattern language structure is one of reasons that patterns stand out from other forms of documentation. The structure of languages makes it possible to define a process for using related patterns in several different sequences.

V. Experiences of Using Pedagogical Patterns in Computer Science

In this section some of the patterns which have been used by the author in different courses in computer science are presented briefly.

You sometimes find that the topics are interrelated in a cyclic way. There are also many occasions where dependencies are acyclic but where students need to know a lot of topics before they can start to solve interesting problems. The pattern Spiral [2] can be used here. It applies to situations where a large number of topics must be mastered at the same time. The suggested solution is to start with an introduction to a subset of the topics without too much detail, but enough to make it possible to solve some interesting problems. For example, you can give the students tools to use for some of the topics they still do not master. You then return to the topics in several cycles during the course. In each cycle the topics are treated more in depth and additional topics may be added.

Everybody make mistakes, not least while learning how to program. Making mistakes give us experiences that help us to learn. So making mistakes while trying to master a subject should be seen as something positive. We know, however, that many students are not confident enough to acknowledge this. They may be afraid of exposing their mistakes. When they make mistakes they also have difficulties in diagnosing the reasons even if their tools provide them with error messages. The patterns Built in Failure [4] and Mistake [3] addresses these problems. The first one emphasizes the need to create an environment where failure is an accepted and even expected outcome of some learning activities. The second suggests giving the student an assignment where they use an artifact e.g., a computer program, with errors and diagnose the errors or interpret the error messages produced by some tool. Another possibility is to ask the students to produce an artifact

with certain errors and study the consequences.

Metaphors are abundant in Computer Science. Well established metaphors include semaphore, client, server, message, editor and desktop. The pattern Consistent Metaphor [2] recommends introducing a new metaphor when you are teaching a topic outside the student's normal experience. A metaphor is useful both for concepts constituting small elements of a course and for giving an overall view of a larger topic.

VI. Discussion

The Pedagogical Pattern concept is an attempt to capture successful practices and for effectively sharing those practices with others. They offer a way to learn from the successes of other teachers. Probably many of the patterns seem trivial for experienced teachers. For novices, however, they can be an efficient way to acquire knowledge and avoid making some of the mistakes that are common for inexperienced teachers. Subjects taught in engineering educations are often developing rapidly and therefore even experienced instructors in these fields often encounter new teaching problems and face new challenges. Pedagogical Patterns give them the possibility to seek advice from other instructors.

A relevant question is, of course, how you know that a certain practice is best. Probably in most cases you don't. The patterns approved by the PPP have been thoroughly discusses, refined and tested. But the forces behind a teaching problem are often conflicting and the suggested solution in a pattern is an attempt to create balance between these.

Some patterns are highly context sensitive since they require special resources or environments. E.g., they may only be applicable when all teaching takes place in small groups or they may assume that schedule changes or small divergences from the original course plan are allowed and can be made on short notice.

Patterns result from lots of feedback and evolve as experience is collected from instructors in different contexts. The project welcomes comments and reactions. There are several other ways in which you can participate: as author, reviewer, test pilot or (as this paper) as an advertiser. Visit the homepage of the project [8] for more information.

References

[1] C. Alexander et al., *A Pattern Language*. New York: Oxford University Press, 1977.

[2] J. Bergin, J. Eckstein, M. L. Manns and E. Wallingford, *Patterns for Gaining Different Perspectives*. Available: http://jerry.cs.uiuc.edu/%7Eplop/plop2001/accepted_submissions/PLoP2001/ewallingford0/PLoP2001_ewallingford0_1.pdf.

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

[4] J. Eckstein, K. Marquardt, M. L. Manns and E. Wallingford, *Patterns for Experimental Learning*. Available:. http://csis.pace.edu/%7Ebergin/patterns/ExperientialLearning.html

[5] J. Eckstein, J. Bergin and H. Sharp, *Patterns for Active Learning*. Available: http://csis.pace.edu/~bergin/patterns/ActiveLearningV24.html.

[6] J. Eckstein, J. Bergin and H. Sharp, *Feedback Patterns*. Available: http://csis.pace.edu/~bergin/patterns/FeedbackPatterns.html

[7] E. Gamma, R. Helm, R. Johnsson and J. Vlissides *Design Patterns: Elements of reusable object-oriented software*. Reading, MA: Addison-Wesley, 1995.

[8] Homepage of the Pedagogical Patterns Project. Available : http://www.pedagogicalpatterns.org/

[9] M.L. Manns, *An investigation into factors affecting the adoption and diffusion of software patterns in industry*. PhD thesis, 2002. Available: http://www.cs.unca.edu/~manns

[10] H. Sharp, M.L. Mann and J. Eckstein, Evolving *Pedagogical Patterns: The Work of The Pedagogical Patterns Projec*t. Computer Science Education, vol. 13, No 4, pp. 315-330, 2003.

You want to introduce a new, abstract topic and you took ABSTRACTION GRAVITY into account.

☛☛☛

In a typical classroom situation students may not know what benefit they might derive from the topic. There is a need to keep students' interest even in abstract concepts.

An abstract concept can become the basis for a large number of applications. However, it is hardly considered useful unless it is related to concrete experience.

☛☛☛

Therefore give the students an example of the problem in a setting that they are comfortable with. After they have found a solution for this example, focus their attention on those aspects that can be applied to similar problems. When your students are inexperienced or you feel that the subject matter is very complex, you should introduce more than one concrete example (see ONE CONCEPT -- SEVERAL IMPLEMENTATIONS).

Use the identified transferable aspects to introduce the general, abstract concept of the solution. When your students have understood the underlying principle, you can advance to a more formal description such as abstractions or patterns.

This kind of presentation is especially useful for students with little or no experience in the course area. It assumes that students are not familiar with the concept with respect to their profession, so that they need to learn a relation that more experienced professional probably already discovered themselves. After some abstractions are introduced this way, the teacher may change the presentation form and start with abstractions before applying it to example situations.

☛☛☛

For example, real life experiences can be used to introduce abstract concepts. When two persons have no language in common, and they do not want to learn another language, they need a translator. Between two existing software systems that do not understand each other, you need a component taking the role similar to a translator. This analogy to a real life experience helps to introduce the concept of the Adapter design pattern that allows establishing contact to a different program without the need to change it.

Fig 1: An example of a Pedagogical Pattern

Participants are overwhelmed by theory	SEE BEFORE HEAR. EXPERIENCE IN THE TINY, SMALL AND LARGE
Participants have problem grasping the whole picture	STUDENT DESIGN SPRINT
Participants don't know how to learn outside the official learning environment.	BUILT IN FAILURE, MISSION IMPOSSIBLE

Fig 2: Part of a quick access table for a Pedagogical Pattern Language