Capturing pedagogical knowledge

Herman Koppelman*

Open University of the Netherlands, P.O. Box 2960, 6401 DL Heerlen, the Netherlands
University of Twente, P.O Box 217, 7500 AE Enschede, the Netherlands

In this paper we argue that much of the pedagogical knowledge of computer science instructors is connected to specific domains, like programming or analysis and design. We explore the possibilities of making such knowledge explicitly available by coupling it to exercises. A pattern-like format is proposed for capturing this knowledge. An example from the domain of building UML class diagrams is elaborated.

Keywords pedagogy; computer science education; patterns

1. Introduction

Novice instructors are usually not profoundly taught how to teach. Typically, a person with specific knowledge that is required will be asked to teach it. But knowing the subject matter is very different from knowing how to teach it. Being an expert in a specific subject matter does not mean that you are good in teaching it.

On the other hand, there are professional, expert instructors who have lifelong experience in teaching. As a consequence they have at their disposal a lot of pedagogical knowledge. So it would be very useful if there were efficient ways to facilitate the sharing of teaching techniques between experts and novice instructors. What is needed is a way of making available in some useful format the collected experiences of the teaching community.

But this is not the end of the story. The explicit availability of pedagogical knowledge is not only useful for those who actually teach in the classroom. It would be of great usefulness if the experiences of skilled instructors in face-to-face education could be made explicit, and be at the disposal of developers and instructors of distance education.

To address the issues of capturing and communicating pedagogical knowledge, pedagogical patterns have been developed. We describe the ideas behind these patterns briefly in section 2.
In this paper we focus on capturing domain specific pedagogical knowledge. Much of the knowledge of instructors is concrete and coupled to specific domains of computer science. In sections 3 and 4 we explore the possibilities of capturing such knowledge, inspired by the way the pedagogical patterns community has been working.

2. Pedagogical patterns

To briefly outline the ideas behind pedagogical patterns, we can do no better than use the words of the pedagogical patterns community itself:

'A pattern is supposed to capture best practice in some domain. Pedagogical patterns try to capture expert knowledge of the practice of teaching. (...) The intent is to capture the essence of the practice in a compact form that can be easily communicated to those who need the knowledge. Presenting this information in a coherent and accessible form can mean the difference between every new instructor needing to relearn what is known by senior faculty and easy transference of knowledge of teaching within the community.' [1] *'For professional educators these patterns might seem obvious, even trivial, because they have used them so often. But for those newer to teaching, these patterns offer a way for experienced teachers to pass on their tried and true techniques.'* [6]

* Corresponding author: e-mail: H.Koppelman@utwente.nl Phone +31 534322894

During the last decade several groups have been collecting pedagogical patterns, and the result is a large collection of useful patterns ([1, 4, 5, 6, 7]). We give a short example, written by Bergin ([4]), about giving feedback.

Positive feedback first.
"You are giving feedback to your student. You have a variety of things to say. You want your students to learn from the feedback you give them and treat it as part of learning.

If you are negative with your students they may tune you out and not listen. If they are especially sensitive they may be hurt. If they are especially arrogant they may take your comments as an attack and attack back. ***You want them to feel good about themselves and also to feel that they can do better****.*

Therefore, when you give feedback, start and end with positive feedback. Suggestions for improvement are sandwiched between these reinforcing comments*.*

Even if you have largely negative things to say, you can still start with the things that were well done and should be retained in the future.

Even in the less positive aspects of your feedback you can take a tone that you are giving suggestions for improvements, not just condemning. You can say "This might be made better if you think about ...", rather than "This is bad." You can also say you don't understand something, or something in a presentation doesn't "work" for you. (...)"

The collected pedagogical patterns are about giving feedback, supporting active learning, and so on, all of them in the context of computer science education. Much less work has been done on capturing pedagogical knowledge that is connected with specific sub domains of computer science. Relevant questions of this kind are for example: how to teach recursion or finding relations for relational databases. In the next section we will have a closer look what this kind of expertise is like.

3. Domain specific pedagogical knowledge

How can the pedagogical domain specific knowledge of instructors be characterized? An important aspect of the work of computer scientists is that they create things: programs, diagrams, specifications, proofs, and so on. Students do not learn to create these artefacts by reading about them or listening to somebody talking about them. As is said in [1]: 'The unexpected difficulties inherent in using the theory or applying the ideas are not likely to be apparent until you actually do use the theory.'

The idea that students learn while being active concurs with the constructivist theory of learning, which claims that knowledge is actively constructed by the student, not passively absorbed from textbooks and lectures. ([2]). So, for many of the topics that are taught in computer science students learn most by performing exercises. And indeed performing such activities is what students actually do in many topics, systems analysis and design and programming in the first place.

In this context the domain specific pedagogical knowledge of professional instructors can be characterized as follows. These instructors, in the ideal case, know the problems the students experience, the common mistakes, the usual misconceptions, the hard modelling questions. They carefully compose or select relevant exercises, that clarify and tackle these problems and from which the students can learn as much as possible. They know these exercises thoroughly and foresee which correct and incorrect solutions the students might conceive. They know how to react as an instructor to incorrect or partially correct solutions. They know which additional questions to ask and which answers to provide. They know which exercises can provoke relevant discussions.

They can control complexity by providing examples and exercises that will give students a valid and complete picture and yet are as simple as possible. Where there is complexity this complexity is essential and leads to good lessons for the students.

They can use different techniques. For example, if instructors aim for a discussion between the students, they ask the students to investigate the quality of some carefully selected artefacts, with some well-known errors, or without errors at all, or without errors but incomplete. Another technique might be to induce students to make well-known errors, in order to discuss these errors explicitly.

So, in the ideal case, skilled instructors foresee the problems of students, choose pedagogically rich examples and exercises to tackle these problems and give adequate feedback to students trying to solve the exercises. In the next section we will explore how we can capture essential parts of this knowledge, in order to facilitate the acquisition of pedagogical skills of those who need them.

4. Capturing domain specific knowledge

This section has 3 parts. Section 4.1 discusses categorizing and describing domain specific pedagogical knowledge. An example of capturing such knowledge is described and discussed in sections 4.2 and 4.3.

4.1 Categorizing and describing pedagogical knowledge

In essence patterns solve problems. Important problems instructors have to deal with while teaching are the problems the students experience when they study the different subjects of computer science. Therefore we categorize the knowledge of instructors according to the problems students experience. Those problems are well-known to skilled instructors.

In the solution part of the patterns, exercises play a central role because they are at the heart of student learning, as we argued in section 3. Moreover, it is possible to link a great deal of domain specific pedagogical knowledge to exercises.

The pedagogical knowledge about the problems of students will be captured in a common format, with the sections *Problem name*, *Context*, *Solution*, *Example*, *Expected answers*, *Questions and discussions* and *Related patterns*. The sections *Problem name* and *Context* are straightforward. The section *Solution* gives a solution in the shape of a description of a type of exercises that can be presented to students in order to tackle the problem. Exercises play a central role in the interaction between student and instructor, as we argued. In the section *Solution* the essential properties of the exercises are described.

The section *Example* describes a concrete exercise, as an instance of the type described in the section *Solution*. The section *Expected answers* describes the type of more or less erroneous or disputable or incomplete answers that can be expected from the students, including appropriate feedback from the instructor to these answers. This section also describes concrete expected answers to the exercise described in the section *Example*.

The section *Questions and discussions* describes the type of additional questions that can be posed and the type of relevant discussions that can be held, in order to deepen the understanding of the students. Alternative answers can be discussed and the merits of different answers can be compared, possible confusion with related concepts can be discussed, the exercise can be extended to include other problems, the effects of slight changes in the exercise can be studied, and so on. This section also describes concrete questions and discussion related to the exercise described in the section *Example*. The section *Related patterns* gives other patterns which are related to this one.

4.2 Example

In this paper we explore problems from the subject matter of building UML class diagrams. According to our experiences building class diagrams is a difficult subject for students. Over and over again we have noticed that students have a lot of problems. They have difficulties with:

- understanding the difference between the concepts class and instance
- distinguishing relevant classes in given requirements
- distinguishing relevant relations between given classes
- fully understanding the concept of generalization
- understanding and applying the concept of association class
- modelling different relations between the same classes

and so on.

We focus on one of those problems. Because the available space is rather limited we have selected a problem with a simple solution. The problem is about the use of a description class.

Problem name
Recognizing the need of a description class.
Context
Often you have a set of objects that share common information but also differ in important ways. An example is the set of copies of the same book in a library. These copies have the same author and isbn, but different bar-code identifiers and they are borrowed by different people. In this case it is useful to introduce a description class (which is also called an abstraction class or a specification class) that contains all the data that are common to the set of objects. It is well-known that novices have problems to see when it is appropriate to use description classes.
Solution
Give the students an exercise where using a description class is appropriate.
Example
A chain of hotels consists of several hotels, in different cities. Each hotel has rooms, with distinct numbers. All hotels have 3 types of rooms: business class, standard and economy. Every type has a fixed standard rate. (These properties are the bare minimum. Others can be added, of course) Figure 1 shows the class diagram.
Expected answers (wrong or disputable)

- Many students do not come up with a description class. In the example they allocate all properties of the rooms to a class HotelRoom, missing the desirability of a class RoomType. In this case the instructor discusses the usual problems of ignoring the description class: risk of inconsistency, inefficiency, possible loss of information.

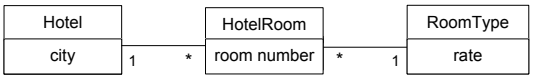

Fig. 1 Class diagram with a description class

- Many students will confuse the use of a description class and generalisation. In the example they will come up with subclasses like BusinessClass, Standard and Economy. The instructor explains that this is not a good idea, because the distinguished subclasses do not have different properties. Crucial is that many instances have exactly the same information. This is an indication that a description class should be used.

Questions and discussions

- The concepts generalisation and using a description class are easily confused. So it is advisable that the instructor provokes a discussion about this confusion, if the students themselves do not do this. To show the difference sharply the instructor can extend the exercise with generalization. In the example a possible extension is: the hotel not only rents hotel rooms but also congress rooms. Congress rooms have a number, like hotel rooms, but they also have a number of places, unlike hotel rooms. Therefore it is appropriate to introduce a super class Room with subclasses HotelRoom and CongressRoom. Such an example clearly shows the difference in a very simple way.
- In many cases using a description class is not strictly necessary. It takes some effort to convince the students why this is desirable and maybe it depends on the persuasiveness of the instructor whether he or she succeeds. In these cases the instructor can extend or change the exercise, in a way that makes the use of a description class inevitable. In the example the exercise can be extended with the reservation of rooms. If a guest makes a reservation for a room, the hotel ini-

tially only makes a reservation for the type, not for an actual room. The actual room is assigned only shortly before the arrival of the guest.

Related patterns

Recently many analysis patterns have been found and published ([3, 8]). The construction described in this pattern is a well-known analysis pattern, called Item-Description pattern or Abstraction-Occurrence pattern. So, this pedagogical pattern suggests the instructor to discuss this analysis pattern.

4.3 Discussion

The knowledge that has been collected in the example pattern consists in the first place of:
- the identification of problems that students experience in a specific sub domain
- the description of pedagogical rich exercises, for tackling those problems
- suggestions for giving feedback to students doing the exercises and for asking meaningful questions.

The example is a simple one. But even simple problems like this one have interesting pedagogical aspects and can provoke relevant discussions, as the example shows.

The formulation of the examples is only provisional: ideas can be tightened up, more knowledge can be added and formulations can be improved. Patterns represent well-known knowledge. They are not invented, but they are found. So we would like to see more instructors involved, as their experiences can be used to review and improve what we have done.

5. Conclusions

Skilled instructors have a lot of domain specific pedagogical knowledge. They know for example the topics novice students experience as difficult and they know how to teach these topics.

In this paper we investigated this knowledge and we showed that substantial parts of this knowledge can be captured in a pattern-like format. This captured knowledge can facilitate the acquisition of pedagogical skills of those who need them.

We showed an example from the sub domain of building UML class diagrams. This example is simple in the sense that only one concept is treated in isolation. But there is no reason not to apply the same approach to more complex problems.

The approach is especially suitable for sub domains where students learn skills by exercising them, such as analysis and design, programming, relational databases and SQL, mathematics for computer science, logic and so on.

It takes a large effort to completely cover sub domains with this domain specific knowledge. Therefore, this approach should be preferably applied to sub domains that are stable, are at the care of computer science and have a firm position in the discipline.

References

[1] Bergin, J. A Pattern Language for Initial Course Design, ACM SIGCSE Bulletin, **33**, 1, 282-286, 2001.
[2] Ben-Ari, M. Constructivism in Computer Science Education. Journal of Computers in Mathematics and Science Teaching **20**(1), 45-73, 2001.
[3] Coad, P. Object-Oriented Patterns. Communications of the ACM, **35**, 9 (1992),152-159.
[4] Eckstein, J., Bergin, J., Sharp, H.C. (eds.). Feedback Pattems, EuroPLoP 2002, http://www.hillside.net/pattems/EuroPLoP2002.
[5] Coad Eckstein, J., Bergin, J., Sharp, H.C. (eds.). Patterns for Active Leaming, PLoP 2002, http://jerry.cs.uiuc.edu/~plop/plop2002.
[6] Eckstein, J., Marquardt, K., Volter, M. (eds.). Patterns for Experiential Learning, EuroPLoP 2001, http://www.hillside.net/patterns/EuroPLoP2001.
[7] Fincher, S., Analysis of Design: An Exploration of Patterns and Pattern Languages for Pedagogy, Journal of Computers in Mathematics and Science Teaching (1999), **18**(3), 331-348.
[8] Fowler, M. Analysis Patterns. Reusable Object Models. Addison-Wesley Longman, Inc. 1997.