Pedagogical Pattern Self Test

(Version 2.0 July 200)

Christoph Steindl
Johannes Kepler University Linz
Linz, Austria
Christoph_Steindl@at.ibm.com

THUMBNAIL

Self tests can be used in any learning situation to engage the students in the learning process. By working through the self tests the students determine themselves whether they have understood the material.

CONTEXT

This pattern addresses the learning process that you have to master as a student. On the first level, you acquire and reproduce knowledge. On the second level, you apply knowledge to solve problems. On the third level, you produce new knowledge. Most teachers are happy if their students can reproduce the knowledge. Industry is happy if problems can be solved, and academia lives from the newly generated knowledge. This pattern concentrates on the first two levels.

PROBLEM

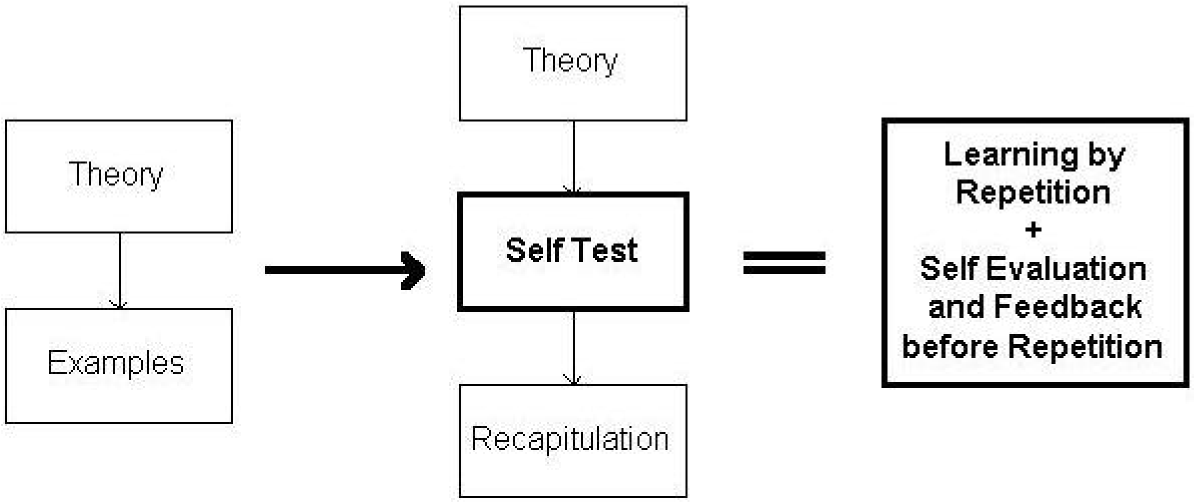
The way knowledge is presented to students is often not engaging or exciting. When the style of lectures resembles a one-way communication, the listeners easily get bored or distracted and lose interest. When they are asked to apply the theory, they fail since they missed some critical points.

FORCES

- There is never enough time to prepare the lecture or to explain the theory.
  ... but both teachers and students want the best learning experience.
- Repetition helps to memorize knowledge.
  ... but repetition gets boring if nothing new is added.
- Learning is fun.
  ... but most academic learning environments are boring.
- Most students don't like to be graded.
  ... but they like to know how they would perform.
- Students dislike exams since exams can contain unanticipated questions.
  ... but exams should be low risk.

SOLUTION

Have the students apply the theory by answering a self test after the theory has been presented the first time. Afterwards the theory can be re-explained, perhaps with some additional details.



First, the students work through the self tests. Then the questions are discussed in a team setting.
Solving a quiz or playing a game can be a way to actively involve students into the learning process.

RESULTING CONTEXT

The self test pattern makes the learning process more efficient by letting the students take an active role: Since the students know that they will have to apply the theory by themselves, they might be encouraged to stay alert during the initial explanation to understand the critical points. Self tests show the students where they failed and what they did not understand. So they will listen more closely to the following repetition of the theory. The important point is that they evaluate their knowledge and get feedback before the repetition.
A drawback is that you (as a teacher) lose some time since the students have to answer the questions. However, this time is well invested: When the answers to the questions are presented, the students will know where they need help.
An advantage is that the lecture becomes much more interactive: the students can ask each other, they can ask the teacher, or work together in small groups. The students are involved actively in the learning process. Additionally, you (the teacher) can walk around the class room while the students answer the questions and look at the answers:

- If the students do not know how to procede, you can point them into the right direction.
- If you spot wrong answers, you can ask them why they think that their answer is right.
- You see who is able to answer the questions and who is not.
  However, you should not make your observations obvious. You have to convince the students that you are not interested in the results and that you will not grade them. The students have to see the self tests as a chance to evaluate their current knowledge.

You must prepare the questionnaires and their answers. However, you can reuse the questionnaires for the examination at the end of the course. The students will appreciate that since they know the kind of questions. You can test how much time the students need to answer the questions. You see which questions are difficult to answer or are ambiguous. You can use this feedback to make the final examination more predictable and reliable for the students and for yourself.
Self tests can become the backbone of a practical lecture with very different styles of questions:

- A question catalogue which summarizes the main points of the lecture. Every student should be able to answer these questions at the end of the semester. These questions make the learning goals of the lecture explicit for both the students and the teacher.

- Practical programming examples where students write a piece of code with paper and pencil.
- Program simulations where students walk through a piece of code, simulate the execution of the program and explain the outcome.
- Open questions where students define and describe newly learnt terms.
- Multiple choice questions where students select the right answer(s).

Self tests can be reused if one teacher leaves the university and another one holds the same lecture or if one lecture is held by several teachers (one prepares the self tests and all use them).

RATIONALE

The best possible knowledge transfer can be reached by communicating the knowledge audio-visually where the students are involved actively and both the teacher and the students have fun.
Knowledge is memorized much better if it is learnt again and again. Simple repetition is, however, less efficient than a process of explanation, application and re-explanation. The key is that the students try to apply the knowledge before it is re-explained.
The students often feel that the theory is either trivial or hard to understand. The issue of the self test pattern is to motivate the students to first listen more carefully during the presentation of the theory, to let them apply the theory and to make them aware of the difficulties before the theory is re-explained. By taking an active role in the learning process, the students overcome their initially passive attitude during the presentation of the theory.

RELATED PATTERNS

The Design-Do-Redo-Redo (DDRR) Pattern and the Design-Implement-Redesign-Reimplement (DIRR) Pattern are about not being satisfied with an initial design, discussing it and finding additional insight by a complete start-over.
The "Spiral Pattern" (by Joe Bergin) is about teaching easy things first and filling in the difficult parts later. Similarly, we propose to present the basic theory first and additional details later.
The motivation behind the "Understanding Through Speaking" Pattern (in "Mining for Nuggests" by Christopher Skelly, private communication) is that if a student (actively) says something, it will leave the room with him. Similarly, we propose that the students (actively) work through the self tests.

KNOWN USES

Text books often follow this pattern: They introduce some concepts, show examples and add some details for the interested reader.

Many teachers hand out old exams (perhaps with a solution) before the exam at the end of the course. Students can test their knowledge by working through the exam.
In "A Berkeley Compendium of Suggestions for Teaching with Excellence" you find the suggestion number 89 to give students problems to solve during class time. The reported procedure is: first outline and discuss a major concept, then give the students a specific short problem and ask them to take 10 minutes to try to apply the new concept.
Application of this suggestion is reported to greatly reinforce the students' learning.

The department for criminal law of the Johannes Kepler University Linz also uses tests (questions and answers) that help the students during the preparation for exams.

The self test pattern has successfully been applied at the Johannes Kepler University in Linz in courses about object-oriented programming (winter semester 1998/99 and 1999/2000, summer semester 2000) and advanced algorithms and data structures (summer semester 1999) by Christoph Steindl, and in introductory courses about software development (winter semester 1999/2000) by Christoph Steindl and several other teachers. The following self tests can be obtained by mailing the author.

Self test 1 (software development 2) (pdf (in German), pdf (in English))	Solution (pdf (in German))
Self test 2 (software development 2) (pdf (in German), pdf (in English))	Solution (pdf (in German))
Self test 1 (software development 1) (pdf (in German))	Solution (pdf (in German))
Self test 5 (software development 1) (pdf (in German))	Solution (pdf (in German))
Self test 1 (algorithms and data structures) (pdf (in German))	Solution (pdf (in German))
Self test 2 (algorithms and data structures) (pdf (in German))	Solution (pdf (in German))
Self test 5 (algorithms and data structures) (pdf (in German))	Solution (pdf (in German))
Self test 1 (object-oriented programming) (pdf (in German))<td	
Self test 3 (object-oriented programming) (pdf (in German))	Solution (pdf (in German))

ACKNOWLEDGEMENTS

Thanks to the participants of the Pattern Writing Workshop in Vienna (October 1999) for the feedback.
I want to express my gratitude to my shepherd Linda Rising for her numerous advices and suggestions for improvement.
Also thanks to the participants of the Writer's Workshop (Pedagogy) of the EuroPLoP Y2K conference .

APPENDIX

Samples for different styles of questions:

- A question catalogue with open questions:
```
1. What is the difference between object-based and object-oriented
   programming?
2. What does "separation of concerns" mean?
3. What does "information hiding" mean?
4. What is the difference between the procedural view of
   operations and the object-oriented view of operations?
5. What does "data abstraction" mean?
6. What is the difference between concrete data types and abstract
   data types?
7. What does "inheritance" mean?
8. What does "dynamic binding" mean?
```
- Practical programming examples:

Implement the constructor *Point(x, y, col)* and the method ***write*** in the following program.

```
      class Point extends basic {
        static final int WHITE = 0;
        static final int RED = 1;
        static final int GREEN = 2;
        static final int BLUE = 3;
        static final int BLACK = 15;
        int x, y;
        int col;
        static int defaultColor = RED;

        Point (int x, int y, int col) { // initializes the point
                                        // with the parameter values



        }

        void write() { // outputs the point textually in the format
        // E.g.: (0,0) in red
        // Known colors are output textually, unknown colors as their
        // number code

        }
      }
```

- Program simulations:
  Simulate the program *TestPoint* and show the expected output within the comments of the main program.

```
      public static void main (String[] arg) {
        Point p1 = new Point(1, 2);
        Point p2 = new Point(2, 1, Point.BLUE);
                                  /* expexted output            */
        equal(p1, p2);            /*                            */
        doSomething(p1);          /*                            */
        equal(p1, TestPoint.p1);  /*                            */
        doSomething(TestPoint.p2); /*                           */
        equal(TestPoint.p1,
              TestPoint.p2);      /*                            */
        TestPoint.p1.write();
        out.writeln();            /*                            */
        TestPoint.p2.write();
        out.writeln();            /*                            */
        p1.write();
        out.writeln();            /*                            */
        p2.write();
        out.writeln();            /*                            */
      }
```

- Multiple choice questions:
  Questions about polymorphism and extension
  1. In principle, polymorphism is the possibility that a variable can refer to several objects at compile time (right / wrong).
  2. In principle, polymorphism is the possibility that a variable can refer to objects of several types (right / wrong).
  3. An extended type T1 can contain more attributes than ist base type T, i.e. T1 is more special than T (right / wrong).
  4. In the same way, the set of T1-objects is bigger than the set of T-objects (right / wrong).