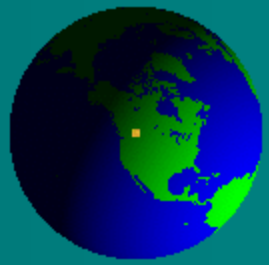
Information Literacy VS Computing (Programming) Literacy - Teachers View

Ljubomir Jerinić

Faculty of Science
Department of Mathematics and Informatics
Chair of Computer Science

Email: ljubomir.jerinic@dmi.uns.ac.rs



Learning Programming

Programming is hard. It's the process of telling a bunch of transistors to do something, where that something may be very clear to us fuzzy humans, with all our built-in pattern matching, language processing, and existing knowledge, but really, horrifically, tediously difficult to communicate to a bunch of dumb transistors.

Dethe Elza

Teaching Programming

From my point of view, few textbooks treat methodological skills of teaching programming appropriately. They teach *programs*, not *programming*. They focus largely on knowledge, not skill. Indeed, I would claim that computer scientists in general do not think much about the programming process and have little idea about how to teach programming.

David Gries
Computer Science
Cornell University

Gries, D., and P. Gries. *Multimedia Introduction to Programming Using Java.* Springer Verlag, NY. 2005.

Problem Statement...

- Learning (teaching) IT and CS is hard – how to learn (teach) IT and / or CS?
- Problem with drop-out from CS and IT courses in USA, Finland, Australia, etc. at the University level – why?
- Where to put learning of elementary knowledge about CS and IT?
- Secondary or Primary level of Education?
- What is the bottom line on knowledge that student have to bring to University?
- OO First?

In Serbia

- In Serbia Educational reform is on 2/3 of long, long way… (?)
- We have done reform at the University education and Primary (?)
- IS and CS are put in Primary education, i.e. the students learn basic of IT from 6-th grade (obligatory course as a part of Technical education) and elementary programming (as the elective course in 6, 7 and 8 grade)
- In Secondary level the reform is stopped so… we don't know what will be.

The Goal Of Research

- To investigate Pedagogical Patterns approach (Pedagogical Patterns Project Home: www.pedagogicalpatterns.org, Bergin, J., Fourteen Pedagogical Patterns. http://csis.pace.edu/~bergin/PedPat1.3.html)
- Try to find out new ones
- And to modify and explore the existing Patterns in a sense of their Instances for teaching Elementary programming

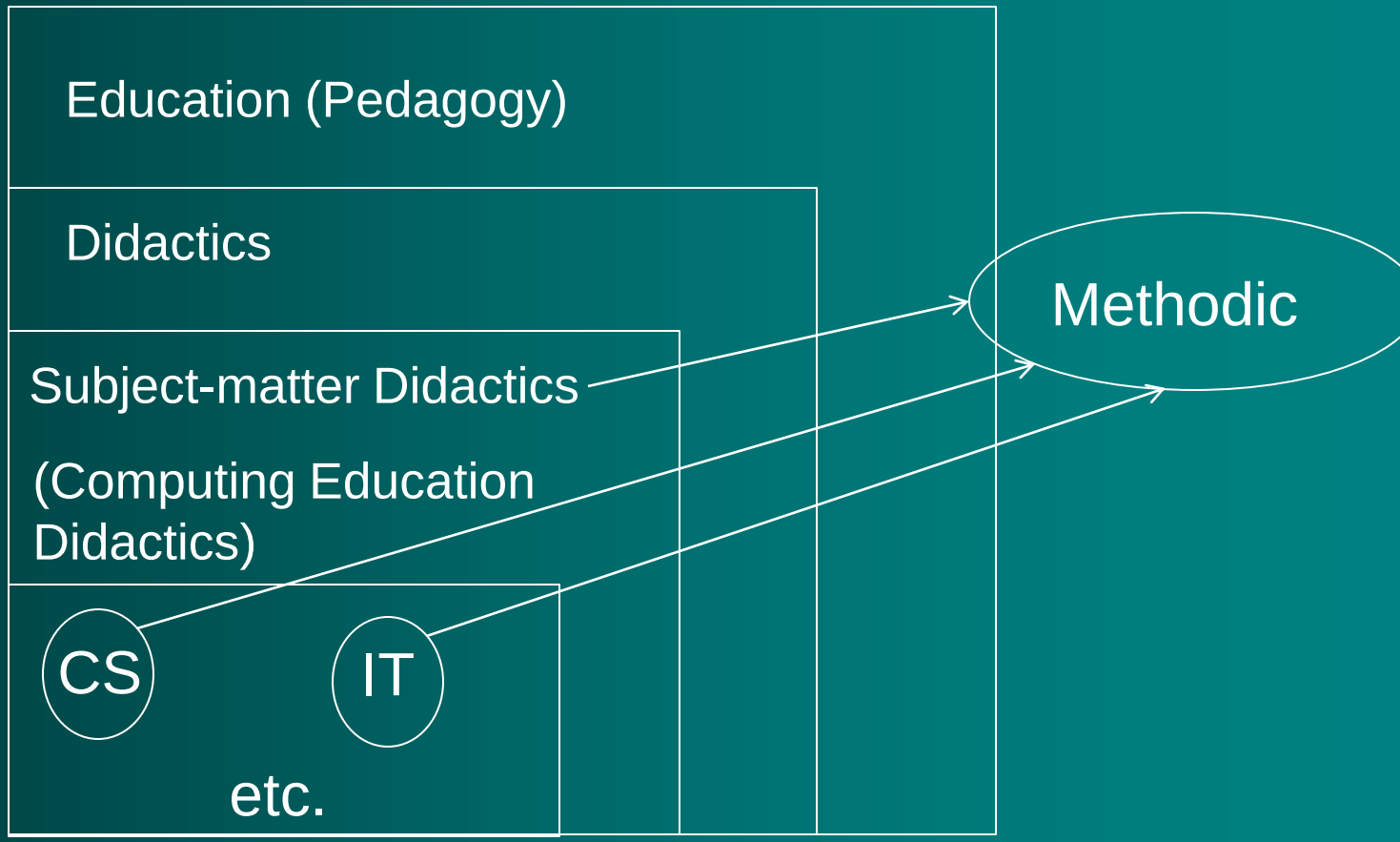
Terminology…

- The American Educational Research Association (AERA) has classified the field of educational research into 12 divisions that represent broad substantives or professional interests.
- However, some fields were not given such an independent status. For example, the philosophy of education did not exist as its own field of a classification title level.
- In addition, continental researchers have understood didactics slightly differently than Anglo-American researchers. In Anglo-American countries didactics was found in under several division titles such as Curriculum Studies, Learning and Instruction, and School Evaluation & Program Development ("12 divisions within AERA", Retrieved January 24, 2011 from http://www.aera.net/divisions/Default.aspx?menu_id=62&id=179).

Terminology…

- The difference in classification and emphasis of subfields in the Continent and the America is a matter of different cultures that have their own philosophical and political roots. Uljens ("On General Education as a Discipline", *Studies in Philosophy and Education*, 20, 291-301. pp. 295, 2001), for example, stated that "*From an American perspective it may seem odd to have several sub-disciplines in education. From a Nordic perspective again it is odd that education is not an autonomous discipline at every American university, but is instead conceived of as a "a field of research"*."
- Methodic ??? (I do not find yet!!!)

So…


Education (Pedagogy)
Didactics
Subject-matter Didactics
(Computing Education Didactics)
CS
IT
etc.
Methodic

Pedagogical Patterns

- Pedagogical patterns, like all patterns, attempt to capture expert practice.
- In this case it is the practice of experienced teachers, both in academia and in industrial settings.
- The pedagogical patterns project [Pedagogical Patterns Project Home: www.pedagogicalpatterns.org] is working on collecting many types of patterns that can help teachers teach and students learn.

Examples…

- Learning by (on) mistakes is very fine teaching techniques or teaching method generally speaking. In teaching Computer Science, Informatics, Information Technologies, and similar disciplines based on technique or technologies, and it is used very often. Joseph Bergin (Fourteen Pedagogical Patterns. http://csis.pace.edu/~bergin/PedPat1.3.html) proposed couple of general Pedagogical Patterns which are directly involved in learning by mistake method of learning, with special implications in usage of them in teaching CS1 and CS2 courses.
  - ***Mistake*** - Students are asked to create an artifact such as a program or design that contains a specific error. Use of this pattern explicitly teaches students how to recognize and fix errors. We ask the student to explicitly make certain errors and then examine the consequences.
  - ***Grade It Again Sam*** - To provide an environment in which students can safely make errors and learn from them, permit them to resubmit previous assignments for reassessment and an improved grade.

Examples…

- A couple of Composite Pedagogical:
  - ***Design-Do-Redo-Redo (DDRR)*** - pattern by Marcelo Jenkins [Pedagogical Pattern #13: Design-Do-Redo-Redo (DDRR) Pattern, http://sol.info.unlp.edu.ar/ppp/pp13.htm], used in teaching Object-Oriented Programming (OOP) to senior students based on a multi-language approach. The idea is to teach OOP concepts such as encapsulation, abstraction, and polymorphism, independently of the OOP language used. To do that, a Design-Do-Redo-Redo (DDRR) pattern is used, in which students design an OOP solution to a programming assignment and then implement it in three different languages. They have to elaborate differences and possible errors.
  - ***Design-Implement-Redesign-Re-implement (DIRR)*** – pattern by Richland, Kornell and Kao [The Pretesting Effect: Do Unsuccessful Retrieval Attempts Enhance Learning? Journal of Experimental Psychology: Applied, 2009, Vol. 15, No. 3, 243–257.]. The pattern could be used to bridge the gap from an old paradigm to a new paradigm (from procedural to object-oriented), emphasizes common programmers mistakes when they tried to “compile” solutions form procedural point of view to object-oriented directly, for example.

My Examples…

- Pedagogical Pattern: Wolf, Wolf, Mistake
  - Topic which is taught is divided into smaller pieces called subtopics or fragments. Fragments are introduced step by step. The goal of the topic is to show usage of these fragments in solving certain problems. After the whole material is presented, some examples of implementation these fragments (or the methods based on them) are shown to the students. They have active participation in constructing the solutions. At the end, an artifact such as a program, object and/or design, with a particular error has been realized. Lecturer knows that mistake is made, but say nothing about that. At the end of the class lecturer just says that all examples have to be tested and verified as homework assignment. Next time, lecturer asks students do they found something in their homework assignments. Lecturer is interested about their opinions on the correctness of the solution that he presented last time. Students should explain the nature and possible consequences of the error, if they were find the mistake at all. Lecturer just conducts the discussion. Using this form, students learn how to recognize specific errors of construction and design, as well as the importance of testing software.

Pedagogical Pattern Language

- **PROBLEM / ISSUE**
- **AUDIENCE / CONTEXT**
- **FORCES**
- **SOLUTION**
- **DISCUSSION / CONSEQUENCES / IMPLEMENTATION**
- **SPECIAL RESOURCES**
- **RELATED PATTERNS**
- **EXAMPLE / INSTANCES**
- **CONTRAINDICATIONS**
- **REFERENCES**

Example of PPL…

- **EXAMPLE / INSTANCES (for** Pedagogical Pattern: Wolf, Wolf, Mistake)
- This pattern could be used effectively in teaching some introductory CS course. If you wish to teach the students about importance of analyzing the boundary cases in program design, and why the testing software is not an easy job, you may use this pattern.
- For example, the pattern was used in Basic of Computer Literacy course for non-professionals (like students with major in Geography) at the University of Novi Sad. Topic on data types and potential problems with them (such as division by zero for numbers, for example) was taught at the beginning of the course. After a while, branching and control structures were done, and their usage in solving some problems is presented. The students together with lecturer solve some problem using these branching and control structures. The lecturer conducted the output. But, the “hidden” special case is not seen by students, i.e. for the particular data entry the program could crushed. They miss to observe the case which leads in dividing by zero. This case lecturer "wisely" ignore in the analysis of the task. Next class, students still did not notice the mistake, and lecturer admitted his “sin”, and explains the reason and consequences of mistake. Couple weeks later, students get the assignment very similar to previously, but in some other context. They all do the assignment without a single mistake.

Feedback

- ----- Original Message -----
  From:* Jutta Eckstein <mailto:jutta-eckstein@t-online.de>
  To:* Ljubomir Jerinic <mailto:jerinicl@eunet.rs>
  Sent:* Thursday, February 10, 2011 10:41
  Subject:* Re: Question
- Hi Ljubomir,
  thanks for sending me your pattern. This looks really good, I think you should send it to one of the PLoP conferences (EuroPLoP, PLoP, or any other xPLoP). The main thing I have been wondering when reading your pattern, was if this pattern doesn't depend on a specific culture? You talk already about students taking the professor's opinion for granted, however e.g. in China or India the professor might even lose his face using this pattern... Maybe you want to add something along those lines to the contraindications or consequences?
  However, your pattern is really up for being presented at a PLoP conference!
  All the best,
  Jutta

Further Work…

- How to Teach Elementary Programming Course (the whole approach)?
- *Spiral* and / or *Semiotic Ladder* are dominant, but are they good enough?
- Approach *Stepwise Improvement* (modification of known *Stepwise Refinement*) could be modified with pattern FINE TUNNING
- Etc. (the choice of first PL, AP Courses, CS0, Scandinavian Model…)

Something Interesting…

Transforming High School Computer Science: CS / 10,000 Project

(For further information, contact Jan Cuny, Program Officer for Broadening Participation in Computing, National Science Foundation (jcuny@nsf.gov, 703-292-8489).**)**

Something Interesting…

If we are to build a globally competitive 21st century workforce and maintain our leadership in IT innovation, there is no stage in the academic pipeline more crucial than high school. It is true that students begin to lose interest in computing much earlier, probably in grades 4-5. Yet engagement programs for middle school students will not be effective if those students have no further opportunities during their four years of high school. Likewise, new and reinvigorated college computing programs cannot have a significant impact if there are too few interested and qualified students to show up at their doors. There are clear indications that college programs are already impacted. Since 2000, the percentage of incoming college freshman who intend to major in computing has decreased more than 70%; for women, the figure is closer to 80%. While some universities believe this trend may be leveling off or even turning around, the HERI data – a survey of incoming college freshman which has been extremely accurate in predicting degree attainment after four years – declined still further in **2008**, **with just 1% of students intending to major in computing.**

Something Interesting…

- The AP CS is a rigorous college preparatory course; however, nationally only 2000 computing teachers have passed the AP audit, indicating that it is being taught in less than 10% of our high schools. Even that course is not optimal: it is programming-centric; it is inaccessible to students with no prior experience; it does not focus on the fundamental concepts of computer science or computational thinking; and it does little to teach the breadth of application or “magic” of computing. Consequently, the **AP CS A test was taken by only 14,529 students in 2008**, as compared to the 204,564 who took the Calculus AB exam, the 141,321 who took the Biology exam, or the 96,282 who took the Statistics exam.
- AP CS A also had the worst gender balance of any of the AP tests. Just 18.3% of the CS AP test takers were women in comparison to the Calculus AB test, where 48.7% were women, or Statistics, where 50.2% were women. Only 11.8% of the AP CS A, test takers were underrepresented minorities. **Clearly, high school computing needs to change.** We are serving too few of our students well. We propose:
  - **CS/10,000 Project Goal***: To develop an effective new high school curriculum for computing, taught in 10,000 high schools by 10,000 well-qualified teachers by 2015.*

The End…

Thank you for your attention