Didaktik

Principper

(citater fra Caspersen, M.E. and Nowack, P.: "Computational Thinking and Practice — A Generic Approach to Computing in Danish High Schools", Proceedings of the 15th Australasian Computing Education Conference, ACE 2013, Adelaide, South Australia, Australia, January 2013, pp. 137-143.)

Learning Activities vs. Knowledge Areas (Learning Goals)

Use-modify-create

"When designing learning activities, we aim at organising the material in such a way that the pupils experience a consume-before-produce progression through the materi- al. Initially, the pupils act as consumers of an artefact by using and studying it; then, they go on to make first sim- ple and then gradually more complex modifications to the artefact. Eventually, the pupils may be requested to build similar artefacts from scratch. The consume-before-produce principle sometimes alternatively characterised as a use-modify-create pro- gression can be applied in many areas. In program- ming, pupils can use programs or program modules before they start making modifications and eventually cre- ate modules or complete programs on their own. The ap- proach applies equally well to other areas, e.g. modelling and interaction design. The origin of (a specialisation of) this principle can be traced back at least to 1990 where Pattis introduced the call-before-write approach to teaching introductory programming (Pattis 1990). In Christensen and Caspersen (2002), the authors apply the principle to provide an al- ternative and incremental way of teaching about software frameworks and event-driven programming in CS1. In Schmolitzky (2005), the author briefly mentions the notion of consuming before producing by providing three specific examples of using the principle in the context of learning object-oriented programming using the BlueJ system (Kölling 2003)."

Application-oriented (outside-in)

"Traditionally, introductory computer science courses apply a bottom-up approach, in the sense that pupils are introduced to basic and foundational concepts and expected to master these before more advanced concepts and principles are introduced. Hence, in a traditional programming course, pupils are often trained in constructing a "Hello World" program as the very first activity, and then later on are trained in adding more layers of complexity to a system in terms of user interfaces, databases, etc. For the technically inclined pupils this may be a feasible approach, but in our case, this could pose severe motivational problems, as we are dealing with a wider range of pupils with much more diverse interests and backgrounds.

There is an even more important reason why a traditional bottom-up approach is fallible. We are not aiming at developing detailed and specific competences in the seven knowledge areas. Overall, we are aiming at developing interest, critical thinking, and broader skills in computational thinking and practice. Therefore we have decided on an application-oriented top-down approach. This means, that we start the various teaching activities by introducing well-known or familiar applications, which we then split apart for conceptual and/or technical examination, evaluation, and modification. For motivational reasons, we choose applications based on the criteria, that they must by themselves be naturally appealing to pupils in our age range. Applications, which they find interesting to use and hopefully to examine and improve. Examples could include pedagogical lightweight versions of Facebook, iTunes/Spotify, YouTube, Twitter, Blogs, Photoshop, and similar applications."

Worked Examples

"A Worked Example (WE), consisting of a problem state- ment and a procedure for solving the problem, is an in- structional device that provides a problem solution for a learner to study (Atkinson et al. 2000, Chi et al. 1989, LeFevre and Dixon 1986). WEs are meant to illustrate how similar problems might be solved, and WEs are ef- fective instructional tools in many programs, including computing. Bennedsen and Caspersen (2004) illustrate implicitly how WEs can be used to teach object-oriented program- ming using a systematic, model-based programming pro- cess. Caspersen & Bennedsen (2007) present an instruc- tional design for an introductory programming course based on thorough use of WE. Caspersen (2007) provides an overview of WE literature related to programming education as well as a survey of the related cognitive load theory. Through didactical training of teachers and systematic enforcement, WE have come to play a key role in the didactical design of most learning activities developed for the new computing subject. A multitude of examples are available from a website maintained by the Danish Asso- ciation of High School Teachers in Computing2. Unfortu- nately, the material is only available in Danish."

Stepwise Improvement

"The Danish Ministry of Education's official guidelines for the new computing subject recommend that all con- structional activities be designed according to Stepwise Improvement. In its original form, stepwise improvement (not to be mixed with stepwise refinement although the two are somewhat related) is presented in the context of program development (Caspersen 2007, Caspersen and Kölling 2009), but the methodology is applicable for the construction of any concrete or abstract artefact. Stepwise improvement is a framework for incremental development of an artefact. According to stepwise im- provement, development takes place in three dimensions: from abstract to concrete, from partial to complete, and from unstructured to structured. Thus, development of an artefact can be characterised as a mixed sequence of re- finements, extensions, and restructurings of the artefact. For the new computing subject, the recommendation from the Ministry of Education is that stepwise improve- ment is used systematically in all constructive learning activities. A number of concrete examples as well as more general guidelines are provided in eight reports pub- lished by the Danish Ministry of Education (2011)."

(se. f.eks. Bartek lave Pong i Scratch: https://www.youtube.com/watch?v=EF_CyoATL0c)