

## **Becoming a programmer requires more than to master a programming language**

**Jens Bennedsen**  
**IT University West**  
**jbb@it-vest.dk**

Programming is a complex task that requires knowledge in different domains. du Boulay (1989) identifies five overlapping domains that the student must master in order to be able to create programs: General orientation (the purpose and possibilities of programs), the notation machine (a model of the execution of a program text), notation (the syntax and semantics of the particular programming language), structures (general problem solving strategies) and pragmatics (the skill of planning, developing, testing).

Traditional programming textbooks and programming courses devote most of their content to presenting knowledge about a particular programming language (Ronins et al 2003), thereby not focusing explicitly on the structures and pragmatics of programming.

One interesting question is why there are almost no focus on the structures and pragmatics. But an even more interesting question is “how can we design our CS1 course so that it has a focus on structure and pragmatics”? In the following I will address this question in four areas: Pedagogy, Progression, Materials and Tools.

### **Pedagogy**

Given that the goal is a focus on pragmatics and structure, one way of achieving this is to expose the students to the way an experienced programmer works. The underlying pedagogy for the course could therefore be apprenticeship based pedagogy as proposed by Dreyfus and Dreyfus (1986) where the students learn by observing, imitating and discussing with the master (teacher). In Nielsen and Kvale two different aspects of apprenticeship based learning is described: Person-centered and de-centered. In the Person-centered case, the student observes and imitates the master, in the de-centered case the students participate in a community of practice. In a particular course one can vary the focus on each aspect; the more novice the lesser possibilities the student has in participating in a community of practice.

### **Progression**

Traditionally progression is based on the concepts of the programming language: First is the simple statements and variables introduced, then types, then inheritance... In Bennedsen and Caspersen another description of progressions is given: progression in the complexity of the underlying class model. This has the nice feature that it is easy to give systematic ways of converting a given class model to the corresponding programming code, thereby learning the students a systematic way of programming by showing them explicit structures and pragmatics.

### **Materials**

The traditional learning materials (textbooks) are static materials. This is problematic in order to show the pragmatics and structures – the programming process. It is therefore needed to use other teaching materials; materials that can capture the dynamics of the programming process. One way of doing this is to use “live programming” showing in the lecture theatre how a program is developed. Another way is to capture this (record what is going on at the screen and the masters voice as (s)he thinks aloud)

## Tools

The novice needs tools that are simple and intuitive to use, not necessarily tools that give a high productivity with respect to lines of code. (S)he also needs tools that give a clear understanding of the notational machine, so an understanding of the relationship between the program text and the execution of the program can be established.

Bennedsen, J. and Caspersen, M (2003) Programming in context: a model-first approach to CS1 Proceedings of the 35th SIGCSE technical symposium on Computer science education Norfolk, Virginia, USA

du Bouley, B (1989). Some difficulties of learning to program. In E. Soloway & J.C. Spohrer (Eds) Studying the novice programmer. Hillsdale, NJ: Lawrence Erlbaum.

Dreyfus, H and Dreyfus, S (1986): *Mind over Machine*. The Free Press New York, NY, USA

Robins, A., Rountree, J. and Rountree N (2003). *Learning and Teaching Programming: A Review and Discussion*, Computer Science Education, Vol. 13, No 2 pp 137 - 172

Nielsen, K. and Kvale, S. (1997). "Current issues of apprenticeship." In: *Nordisk Pedagogik* 17: 130 - 139.