Programming for novices: adequacy between concepts, learning tools and novices' needs

Henry Julie
NAmur Digital Institute
University of Namur
Grandgagnage street 21
Namur 5000, Belgium
julie.henry@unamur.be

ABSTRACT

For more than 30 years, the difficulties encountered by novices in programming have involved computer scientists, psychologists and pedagogues, among others. There are many ways to explain these difficulties: individual differences (who), concepts (what), methods (how) and tools to aid learning (with what). This research aims at taking full advantage of this diversity, in particular with a study on the adequacy between concepts of programming, novices' needs and tools to assist learners.

Categories and Subject Descriptors

Human- centered computing → Human computer interaction (HCI) → **Empirical studies in HCI ;** Social and professional topics → **User characteristics - Computer science education**

General Terms

Human-centered computing ; Social and professional topics

Keywords

novice programmers, individual differences, computer science education, cognitive profile

1. CONTEXT AND MOTIVATION

For decades, the teaching of programming to novices has been the subject of numerous studies on topics as varied as the curriculum and associated didactics (what to teach), pedagogy (how to teach it), languages (through what) and tools to aid learning (with what) [10]. However, these researches had limited effects on classroom practice. One reason may be that these are often a solution for a local problem, specific to an institution or a course. They therefore do not meet the needs of other institutions or teachers.

The difficulties encountered by students to learn programming, unanimously emphasized by the literature [11,13], are still a major problem. Indeed, the teaching of programming is about to be reintroduced within a few years into the fundamental and secondary school curricula in the French-speaking part of Belgium. For several years, the French Community of Belgium has equipped schools with technologies (computers, tablets) and, since 2016, *Arduino* and *Thymio*[1] are part of the proposed catalog.

Using microcomputers and robots is not a new approach [9] and does not provide a complete answer to the challenge of teaching programming. Numerous studies show little evidence of a real contribution to learning (and therefore understanding) of programming. Commitment and collaboration are often results observed in children, a preferred target audience when using robots. Fortunately, robots are not the only tools to aid learning. These have multiplied and diversified to meet different needs and audiences [7,14]. However, it is regrettable that the tools are often exclusive in many case studies: a unique tool (for a naturally equally unique context) that aids to learn all the core concepts of programming.

Perhaps learning of programming could be optimized by using a different learning tool for each concept. Perhaps learning of programming could be optimized by proposing to children to make a choice in different learning tools.

2. BACKGROUND

The first impact of differences between individuals in the field of Human Computer Interfaces (HCI) was measured almost 30 years ago. The results concerned, among others, programming [2]. It was then necessary to adapt the interfaces, and more generally the systems, to the users: "everyone should be computer literate" became "computers should be user literate".

But what do individual differences mean? One of the earliest classifications of characteristics that can affect the use of a computer [1] refers to cognitive styles, personality factors, psycho-motor skills, experience, goals and needs, expectations, preferences, cognitive strategies and abilities. A large number of individual characteristics have been mentioned in the HCI literature, but the terminology used and the relationships between them are not always clear. A categorization has been proposed [6]: "personal user characteristics", "prior acquired knowledge and skills" and "system related user characteristics". For each of these categories, the authors identify characteristics, define them, exemplify their implementation and evaluate their relevance from the adaptation's point of view. This categorization could be a reasonable starting point for research that takes into account the differences of each individual in learning programming. Indeed, if the relationship between individual characteristics, particularly cognitive styles, and learning programming is no longer to be

[1] https://www.thymio.org/

proved [4,8,12] **why not play this card and provide education offering everyone the opportunity to learn through the tool that it is the most suitable?**

3. RESEARCH GOALS & METHODS

The aim of this project is to study the adequacy between concepts of programming, novices' needs and tools to aid them learning programming.

Thanks to research carried out directly in the field, we hope to achieve this goal through several objectives:

- Define the needs of novices in relation to their profile (based on development and cognitive styles)

- Identify the tool(s) most suited to the learning of a core concept in programming

- Identify the tool (s) best suited to a particular student profile
- Measure the adequacy between the three components: core concepts, learning tools and student needs

The research will consist of two main phases: a first (already in progress) with an "education" orientation, focusing specifically on the teaching of programming, and a second with a "HCI" orientation, focusing on the creation and the evaluation of prototypes of tools to assist learning of programming.
Given the importance of the actors (novice students in the first year of bachelor), the research is based on a method allowing to understand these actors: the Grounded Theory Method [5].

4. DISSERTATION STATUS

Observations were made during two years with students in the first year of bachelor following the Introductory course to programming. Tests are passed by students to validate the existence of predictors of their final examination results. Interviews are carried out in order to explain these results.

A first review of the literature around tools to aid learning programming highlighted the value of tangible tools. A classification of these tools according to Fishkin [3] was then initiated. These tools have also been classified according to the taxonomy of Kelleher [7], notably highlighting the need to increase this taxonomy.

Very few researches about tangible tools are conducted with students over 18 years-old (target audience of this research). An experiment is therefore planned during the academic year 2017-2018: teaching, in addition to a lecture, some core concepts in programming (variables, conditional statements and loops) to first-year students with micro:bit[2] and observe the consequences of this "tangibility" on the students' mental model.

Future work should include comparison with other (not necessarily tangible) learning aids for these same core concepts. Original prototypes will then be developed, inspired by the lessons learned from the different experiments set up and oriented towards more advanced concepts (scope of variables, parameters of a function or return value).

5. EXPECTED CONTRIBUTIONS

According to the research, we hope to produce, besides scientific publications describing the progress of the project:
- A methodological aid for teaching / learning programming according to the student's cognitive profile: what are the needs of the novices?
- A taxonomy, inspired by [7], of tools to aid learning programming (with highlighting tangible tools).
- A review of the tools with a measured impact (and a description of this impact) on learning core concepts in programming.
- An aid in the design of a tool for teaching programming according to the student's cognitive profile and / or according to the concept addressed: what should be the characteristics of this tool?
- Prototypes (including tangible prototypes) that assist in the teaching / learning of programming and the reporting of their evaluation by users.

6. REFERENCES

[1] Browne, D., Norman M. and Rithes D. (1990). “Why build adaptive systems”. In: Adaptive user interfaces, pp. 15–59.

[2] Egan, D. (1988). “Individual differences in Human-Computer Interaction”. In: Handbook of Human-computer Interaction. Ed. by M Helander. Elsevier Science B.V. Publishers, pp. 543–568.

[3] Fishkin, K. P. (2004). A taxonomy for and analysis of tangible interfaces. Personal and Ubiquitous Computing, 8(5), 347-358.

[4] Gilberto, C.-R. et al. (2015). “Methodology for characterization of cognitive activities when solving programming problems of an algorithmic nature”. In: Olympiads in Informatics, p. 27.

[5] Glaser, B.G. and Strauss A.L. (2009). The discovery of grounded theory: Strategies for qualitative research. Transaction publishers.

[6] Granić, A. and Nakić, N. (2010). “Enhancing the learning experience: Preliminary framework for user individual differences”. In: Symposium of the Austrian HCI and Usability Engineering Group. Springer, pp. 384–399.

[7] Kelleher, C. and Pausch, R. (2005). “Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers”. In: ACM Computing Surveys (CSUR) 37.2, pp. 83–137.

[8] Mancy, R. and Reid N. (2004). “Aspects of cognitive style and programming”. In: Proceedings of the Sixteenth Annual Workshop of the Psychology of Programming Interest Group, pp. 1–9.

[9] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

[10] Pears, A. et al. (2007). “A survey of literature on the teaching of introductory programming”. In: ACM SIGCSE Bulletin 39.4, pp. 204–223.

[11] Piteira, M. and Costa, C. (2012). “Computer programming and novice programmers”. In: Proceedings of the Workshop on Information Systems and Design of Communication. ACM, pp. 51–53.

[12] Renumol, V.G.,Janakiram, D. and Jayaprakash S. (2010). “Identification of cognitive processes of effective and ineffective students during computer programming”. In: ACM Transactions on Computing Education (TOCE) 10.3, p. 10.

[13] Robins, A., Rountree J. and Rountree N. (2003). “Learning and teaching programming: A review and discussion”. In: Computer science education 13.2, pp. 137–172.

[14] Sorva, J., Karavirta, V. and Malmi L. (2013). “A review of generic program visualization systems for introductory programming education”. In: ACM Transactions on Computing Education (TOCE) 13.4, p. 15.

[2] http://microbit.org/