Non-Traditional Novices' Perceptions of Learning to Program: A Framework of Developing Mental Models

Thomas Hvid Spangsberg
Digital Design and Information Studies
Aarhus University
Aarhus, Denmark
tbhs@cc.au.dk

Sally Fincher
School of Computing
University of Kent
Canterbury, United Kingdom
s.a.fincher@kent.ac.uk

Sebastian Dziallas
School of Computing
University of Kent
Canterbury, United Kingdom
sd485@kent.ac.uk

***Abstract*—In this research full paper, we present a study of a cohort of non-traditional students learning to program as part of an intercalated computer science year named Year in Computing at University of Kent, Canterbury in the UK. The study uses focus group interviews with students to explore their articulations and perceptions of learning to program. During the interviews, we used Bebras puzzles as a tool for students to reflect on their own experiences in learning to program. By observing the students working with the puzzle, some of the tacit information of problem-solving strategies became apparent. We see Bebras puzzles as a way to gain valuable insights that are not as easily available when relying on conventional self-reflection in a qualitative study. Another contribution grounded in this study is the proposal of a framework describing the students' developing mental models in the beginning of their studies. The results presented here are a first iteration in an ongoing endeavour of exploring students' development of mental models. We hope that these contributions will provide teachers and researchers with a new perspective for developing introductory programming curricula and to engage in further development of our framework.**

Keywords—non-traditional novices; bebras puzzles; introductory programming education; non-CS majors.

I. INTRODUCTION

Supporting students' development of mental models is a crucial aspect in building a deeper understanding of the topics taught [1]. Many articulations of mental models and their applications have been put forward as predictors of success in learning to program [1-5]. In this paper, a framework for articulating students' development of mental models is proposed. We also report on the experiences of using Bebras puzzles [6] as a qualitative research tool for uncovering students' tacit learning experiences. These puzzles present problems that are frequently encountered in computing education, but do not rely on any prior computing knowledge. When used in this context they are able to act as triggers for students to demonstrate their knowledge and skills gained rather than relying on self-reflection. The method of analysis used in the study is a grounded theory approach [7-10] which produced a subjective theory suitable for the context in which it was developed.

This study is financially supported by the Centre for Computational Thinking and Design, Aarhus University in Denmark and IT-Vest in Denmark

II. RELATED WORK

A. Mental Models and The Notional Machine

Mental models can be generally described as "*conceptual organizations of information in memory*" [1]. Mental models can also be explained can be explained as structured knowledge in a student's mind that enables problem solving and abstraction [1]. This connects to students' ability to visualise and simulate outcomes in their mind. Mental models in general are difficult to define because of their abstract nature. Apart from being abstract, they are also dynamic and changes over time, which contributes to the difficulty of defining particular mental models [1]. This also means that there will be many different mental models present in a certain learning situation. Studies performed by Dehnadi [5] identified 11 different mental models in play with a group of 60 students working on the same problem. In this sense, mental models seem to be difficult to work with as they need to be discovered individually when studied. Nevertheless, there is a specific kind of mental model often referred to in the context of computing education: the "notional machine". The notional machine is an abstraction of the execution of a computer program formed in the mind of the novice learner. It is an articulation of how the learner describes the computer interprets and executes a program. Benedict du Boulay first articulated the concept of the notional machine in 1986 [11]. He defines it in this way: "*The notional machine is an idealized, conceptual computer whose properties are implied by the constructs in the programming language employed*." [12]. As with any other mental model, there will be several notional machines in play when students learn to program. As the learner begins to build more and more elaborate programs in the language taught (or, indeed, begins to learn more languages), several notional machines may be constructed and developed [11]. Simple examples range from the anthropomorphic idea that there is an imp (or other intelligence) inside the computer, interpreting human instructions, to understandings of deterministic instructions and flow of control. In this way, the notional machine becomes a mental model of the real machine formed through the

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

programming language. The notional machine is simply a term used to label mental models concerned with a student learning to program – specifically said students' understanding of the computers inner workings when programmed.

B. Spatial Puzzles in Computing Education

Spatial abilities have been considered a predictor of success in the STEM domain for more than 50 years [13]. In computing education research, several studies have proposed a relationship between spatial skills and computing abilities [14]-[18]. Miller et al. [14] explore how the LOGO programming language has an impact on fifth- and sixth grade students' problem solving and spatial abilities. LOGO is widely known for its use of turtle graphics, in which commands for movement and drawing produce line graphics either on screen or with a small robot called a "turtle". Miller et al. found that students who were given instructions in the LOGO programming language outperformed those who were not in their ability to solve problems and, partially, in spatial abilities.

Jones and Burnett [15, 16] argue that there is a relationship between spatial abilities and programming success. One of their studies [15] is concerned with students' navigation of source code based on their mental models of the program in question. Jones and Burnett report that students with low spatial abilities took longer to achieve an understanding of the source code and the pattern of their navigation of the source code differed: students with high spatial abilities jumped between functions more frequently. Another study [16] showed a correlation between mental rotation abilities and the ability to learn programming. More recently, Cooper et al. [17] presented a study in which students were given training in spatial abilities as part of an intensive programming course. These students performed better than students who received no such training despite having taken time out of the course to do the spatial skill training, resulting in lesser instruction in programming than the control group [17].

There are several other studies that use spatial skill puzzles to show a relationship between spatial abilities and learning to program [18-21]. These studies share a common data set and use the same spatial puzzle: a paper-folding test presented by Ekstrom et al. [22] (see Fig. 1) that gauges spatial ability by asking the test subject to predict a pattern which would result from punching a hole in a piece of folded paper.

Other studies have been concerned with differences in spatial ability between men and women. Even though there are not general differences in the general intellect between the sexes, males score significantly higher in spatial abilities compared to females [23]. Reiley et al. [23] presents a comprehensive literature review on the subject and also notes that: *"(…) spatial proficiency can be improved through relatively brief interventions."*. Also, there is no evidence suggesting that there is any inherent difference in performance between the sexes when it comes to computing given equal opportunities and training. As Kafai [24] notes, there are *"(…) no significant differences between boys' and girls' programming performance and interest"*.

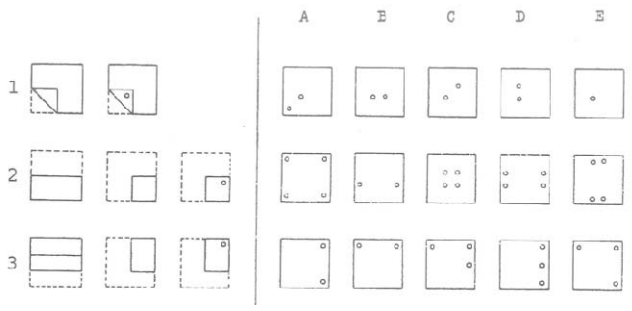

Fig. 1. Paper Folding, based on Ekstrom et al. [22]

III. METHODOLOGY

The study presented in this paper is grounded in data collected during one term of the Year in Computing at the School of Computing at University of Kent in the UK. The programme is an intercalated 120-credit year as part of which students study a specially-designed computing programme taught separately from the regular Computer Science undergraduate degrees. It is taken as part of an existing undergraduate degree (in any subject other than Computing). It does not contribute to degree classification, which depends on the "home" discipline alone, but module marks appear on transcripts. The cohort consists of 35 students with backgrounds from all disciplines (STEM, social sciences and humanities) and of which 40% are women.

The focus for the study was the introductory programming course taught in the first term of the programme from October 2017 until December 2017. This course gives a general introduction to programming and the web through JavaScript, HTML/CSS and basic database manipulations using PHP and MySQL. The data presented in the next section was gathered from three semi-structured interviews over the course of the academic term with eight students who were recruited on a voluntary basis. (All eight students were women.) The first interview was conducted in the beginning of the term and had five participants in a focus-group interview. The second interview was conducted mid-term and had two participants in a second focus-group interview. The third interview was conducted with one participant at the end of the term as a one-on-one interview. There was no overlap in people taking part in the different interviews.

During the interviews, two Bebras puzzles were given to the students to solve together. The first puzzle, "Spinning Toy" [25] (see Fig. 2), was of a spatial nature, where the students would determine a sequence of left- and right turns to get a ball to fall out of a labyrinth mounted on a central pivot.

The second puzzle "Drawing Stars" [26] (see Fig. 3) was concerned with a system for labelling stars using two numbers in this syntax: X:Y. The first number (X) represents the number of points in the star, while the second number (Y) indicates "*if a line from a dot is drawn to the nearest dot (the*

number is 1), the second closest dot (the number is 2), etc." [26] In addition to the spatial dimension, this puzzle also presented the students with a pattern to learn and to apply. The audio of the interviews was recorded and subsequently transcribed.

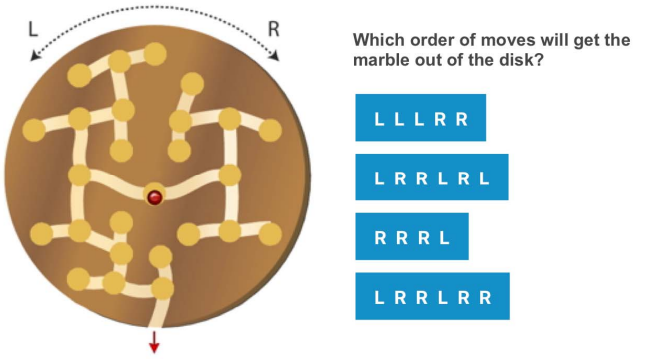

Fig. 2. Spinning Toy Bebras Puzzle

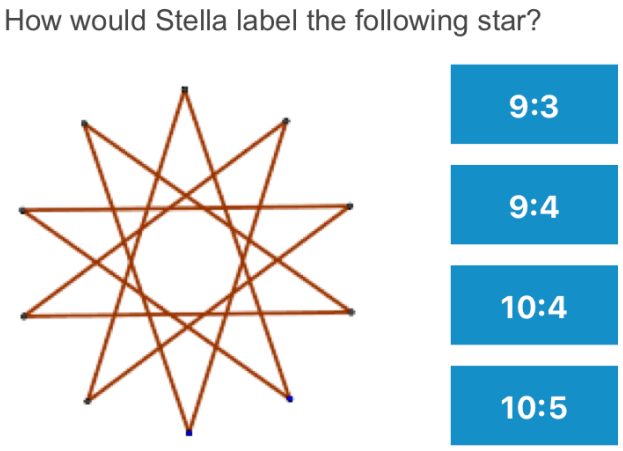

Fig. 3. Drawing Stars Bebras Puzzle

A. *Grounded Theory and Substantive Theory*

We chose to analyze the transcripts using a grounded theory approach, as one aim of this work was to discover the students' experience, not to make a theoretical guess at their states of mind. Grounded theory can be understood as an inductive approach to hermeneutics [9]. Where traditional research approaches seek to move from the inductive to the deductive mode, grounded theory revolves around the two modes in a circular pattern in order to build theory where there might not be existing theory available and is therefore often associated with explorative research endeavors [7, 10]. It is possible to develop general theory through grounded theory methods, but it is often aimed at substantive theory [8]. Substantive theory is developed within one field and is not generally applicable to other fields of study. Rather, there may be transferrable aspects of the developed theory that might be used as a springboard for developing general theory [8]. We open-coded the transcripts in two passes with the goal of forming themes as a basis for the final analysis of the data. We initially identified seven themes, which were reduced to the four we present here based on overlap and redundancy. Regarding the saturation level of the data, it is considered to be satisfactory as all of the interviews complement the articulations made but none of them add significantly new insights when compared to each other.

B. *Limitations*

Only the first and third interviews were suitable for transcription. The audio quality of the second interview was too poor for verbatim transcription. Data from this interview was instead represented as interviewer notes and observation and summaries from the recordings. Therefore, there are no direct quotes from this interview in the results section.

Not all participants were equally active during the interviews. Where appropriate, background acknowledgements are noted in the transcripts.

The size of the groups varied between interviews and the last interview only had one participant. The fact that the interviews were conducted throughout the term also makes comparisons problematic, as the students were interviewed on different stages of their studies.

All participants were female, although this was not a deliberate choice by the authors as the students volunteered to participate in the study. The main cause for these limitations may then be traced back to the sampling method. If another method had been employed for the solicitation, it could have ensured both a higher number of participants and diversity.

IV. THEMES

In the following section, we present the four themes in detail, which emerged from our data. They are named as follows: Learning-Processes and Skills, Puzzle Solving and The Relation to Computing, Struggles and Identity.

A. *Learning – Processes and Skills*

When asked to reflect on their own learning during the programming course, students articulated both skills and processes related to their learning experience. The students mention "attention to detail" as an important skill when it comes to learning programming.

"Like you [Participant 1] said already ***attention to details*** *is a big one."*

[Participant 2, Interview 1, 22:07 mins]

The words “logic” and “structure” also appear frequently during the interviews.

“Logical reasoning [Acknowledgements in the background] (...) especially with the JavaScript assessments (...) So I guess it's ***sequential thinking as well as logical thinking*** *(...)”*

[Participant 2, Interview 1, 23:16 mins]

“I did studies in linguistics as part of my degree, there's some ***similarity there in using logic*** *but I do think it's very different.”*

[Participant 1, Interview 3, 01:31 mins]

“(...) having a ***logic*** *of what I need to use for what (...)”*

[Participant 1, Interview 3, 17:57 mins]

Perseverance can also be traced among the students.

“I just ***keep rewriting the code until I get the result that I want****.”*

[Participant 2, Interview 1, 34:06 mins]

They also give an insight into how their strategies for when they got stuck developed throughout the course.

“I just ***Google what's being taught that day*** *and get a basic understanding of it.”*

[Participant 2, Interview 1, 07:38 mins]

“I don't know sort of ***googling by myself****. But then also sort of* ***trying to think in what ways*** *[Indistinguishable****] searching those keywords*** *of the part of the code that don't work.”*

[Participant 1, Interview 3, 21:26 mins]

“The thing that comes up isn't exactly what I want. Then I just read that and it is not quite what I want ... But I've used at least ***some key words that they are using*** *- then this might be closer to what I want. So,* ***I search again****.”*

[Participant 2, Interview 1, 09:35 mins]

Some students then engage in a form of “hermeneutic googling” – they start by looking up a few keywords or posing a question in the search bar and, based on the results, continue to search using phrases and keywords from the first search until they arrive at a satisfactory understanding of a topic or a solution to solve a problem in their code.

The participants also rely on each other to deepen their understanding of the materials taught.

“Lots of us definitely ***help each other****.”*

[Participant 1, Interview 1, 00:39 mins]

“I definitely find ***helping each other in class*** *if she understands something in her way and* ***explain it to me*** *in a non-computing way.”*

[Participant 1, Interview 1, 01:05 mins]

B. Puzzle Solving and The Relation to Computing

This theme is concerned with how the students solved the Bebras puzzles they were given as part of the interview. It was possible to propose an abstraction of the strategy used as follows:

1. Read task out loud (or internally) from the paper given.
2. Establish common understanding – exploring examples, doing trials.
3. Take a step (Spinning Toy) || Establish X of X:Y (Drawing Stars).
4. Evaluate – eliminate solution based on action done.
5. Take next step (Spinning Toy) || Establish Y of X:Y (Drawing Stars).
6. Evaluate – eliminate solution (if possible) based on action done.
    a. Star puzzle done
7. Repeat 5 and 6 (Spinning Toy) until solution was found.

The following articulations are some of the students’ reflections on how solving the puzzles related to their experience of learning to program.

*“****Attention to detail****. Like in javascript if you forget like a curly bracket and your code doesn't work you have to go back through it and see where you didn't put that in. I believe it's a little bit the same with the quizzes as well, you have to look through it to see where you went wrong.”*

[Participant 1, Interview 1, 19:43 mins]

“(...) there is something like ***the structure that you follow with JavaScript*** *and I feel like with these puzzles. That's also important that you need to find some* ***logic*** *or some sense on how to apply it.”*

[Participant 3, Interview 1, 20:14 mins]

*"It's the same non-verbal **reasoning** we use to kind of figure out in JavaScript (...)"*

[Participant 1, Interview 3, 12:38 mins]

C. Struggles

Students also mentioned that the process of translating a written task into something that they can program is a struggle.

*"To be honest, that's **the problem I struggle with** ... Once I figure out what I want, to make the **jump to 'this is the code components I need'** (...) It's the **way of thinking** that's really difficult."*

[Participant 1, Interview 3, 12:53 mins]

*"It's more the English and the **language barrier** [of the programming language] that you have to go through to think about the loops"*

[Participant 1, Interview 1, 32:59 mins]

*"**Breaking the problem into bits, yeah**."*

[Participant 1, Interview 3, 27:00 mins]

The students also seem to struggle with the amount of new material introduced every week: they feel they lack the time to keep up with the "learning curve".

*"It feels a bit **like doing assignments just to pass the test** rather than doing assignment to learn it because they're **so close together**."*

[Participant 1, Interview 3, 19:15 mins]

*"Trying **to understand everything** as good as possible. [Participant 1]. Especially because at every lesson there is **something new so you need to keep up** [Participant 3]."*

[Participant 1 and Participant 3, Interview 1, 26:39 mins]

Further articulations provide some insight into what the students mean by the word "logic", which has appeared several times.

*"**Maths**, probably - just **the logic aspect** of it."*

[Participant 1, Interview 3, 02:02 mins]

*"I guess you kind of use your **language** when you go through things like for example a for-loop."*

[Participant 1, Interview 1, 32:59 mins]

*"I think sometimes the **math** can come in place when you want to **check it manually** yourself and see that the code you've written, even though it works, **that it returns what you want it to return."***

[Participant 3, Interview 1, 33:28 mins]

D. Identity

The final theme is concerned with students' identity. There seems to be a strong and shared identity amongst the students. It seems to be connected to them having many of the same struggles.

*"**All on the same boat.**"*

[Participant 2, Interview 1, 00:42 mins]

*"So, it has definitely been helpful having people around. And **we're all in the same boat so we might as well ride it together"***

[Participant 2, Interview 1, 02:26 mins]

There is also an indication of an ironic distance to the new skills and abilities learned:

*"(...) one of my friends and he was asking me like, 'I don't understand why this, something, something, something.' And I was like "It's something like this ..." and **I was using all the words** and after I was done with the conversation I was like **'Who am I??' [Laughter] 'What have I become'**."*

[Participant 2, Interview 1, 34:59 mins]

V. Analysis

The purpose of this study was twofold: We wanted to explore students' development of mental models. In doing this, we wanted to see how Bebras puzzles could be used as a tool for examining students' articulations of tacit experiences when learning to program.

A. Framework of Students' Mental Models

We observed three distinct activities:

a) "hermeneutic googling", the struggle of translating the task into something the student knows how to build in code,

b) their use of the word "logic" in several contexts, and

c) the notion of "(…) *keep rewriting the code until I get the result that I want.*" "[Participant 2, Interview 1, 34:06]

These can be combined into a diagram of students' mental models as represented in Fig. 4. The framework should be read as states, not as levels of development.

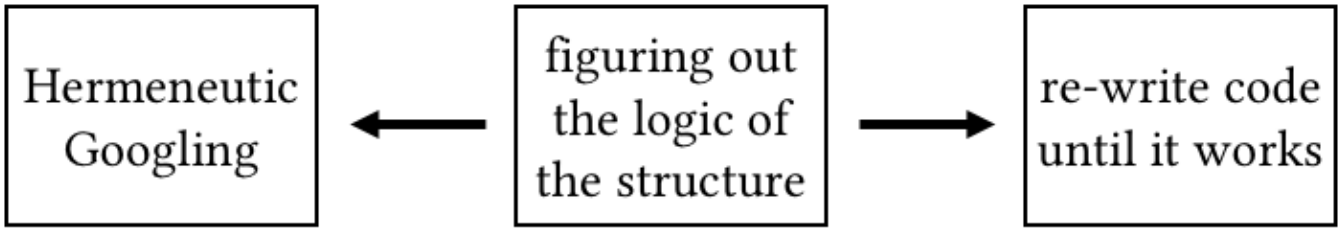

Fig. 4. Framework of Students' Mental Models

At the far left, "hermeneutic googling" is concerned with building a vocabulary for solving problems on a conceptual level. Here, the problem of translation can be seen as a main concern as the student is trying to figure out how to express the problem at hand in order to get help from online resources. This state may come into play when the student is faced with a new programmatic concept different from the ones already mastered. At the far right, re-writing code until it works is concerned with the construction of code for solving a given problem. But at this end of the spectrum, the student is simply trying to rearrange and put together snippets of code in the hope that the final program behaves as intended. We interpret this behaviour as a lack of a notional machine, as they appear not to understand the purpose of their program's organisation and are just moving tokens around in an algebraic fashion. In the middle, the mental model is concerned with comprehension of the components of the programming language in order to solve a certain problem. We interpret the dynamic between these states as a notional machine under construction. It is most apparent in the mid-state of "figuring out the logic of the structure". The articulation implies an awareness of a structure and also that there is a certain "logic" to it. The word "logic" here is a key component in our interpretation of this state. It indicates, that the student is formulating connections between knowledge of different parts of the structure, hence actively developing a notional machine. While the rearranging of code in the far-right state does not contribute directly to this, when some problem has been solved it is possible for the student to revisit the solution in the middle state and examine why the code now works. The far-left state enables the student to articulate this with the vocabulary in place at the moment.

The key to employing this framework in classes is to help students' transition into the middle state through the activities they are engaged in. This means that teaching material should support the transition and take into consideration that students following the same class might not have "achieved" the same state of their mental model. This framework is merely descriptive and is appropriate (and limited) to the research methodology we employed. It is the product of a subjective theory that fits the context in which it was developed. As such, it characterises the mental models developing in these non-traditional novices.

B. Insights Gained from Bebras Puzzles

The process the students used to solve the Bebras puzzles, resembles an "isolation" debugging strategy where "*debugging is viewed as an iterated process of developing bug hypotheses and verifying or refuting them through the analysis of the program behavior*" [27]. Steps 5-7 in the abstracted strategy presented in section IV.B are significant in this context. In the puzzles we used, the "bug hypotheses" [27] is analogous to the different possible answers in each puzzle, "verifying or refuting them" [27] can be observed when the students would look for possible solutions to exclude in step 6.

All of our participants systematically adopted variations of this strategy. We hypothesise that this a sign of students' internalisation of their experiences with constructing code and re-writing it when it does not work as intended – and thus of developing a notional machine. As the theme of "struggles" presented in section IV.C indicates, the students view the construction of code as a challenge in terms of translating the problem into the components needed to be coded. Here, they consider "attention to detail" and the structural aspects of programming to be important aspects. The data does not give any hints towards how deep the student's comprehension of programming is, but it may be higher than the students are consciously aware of. The seemingly internalised debugging strategy was not something the students themselves articulated but something they rather demonstrated during the puzzle-solving part of the interview.

VI. Conclusion

In this study, we explored the use of Bebras puzzles as a way to examine students' tacit learning experiences. We used a grounded-theory approach to construct a subjective theory that shows a spectrum of stages of development in learning to program that students presented in this context. The construction of a notional machine is closely connected but does not cover the full spectrum of the articulations made by the students. The use of Bebras puzzles in the data gathering process seems to have uncovered a systematic problem-

solving process similar to a debugging strategy of isolation. This process would not have been apparent without the use of the Bebras puzzles.

We hope that these contributions will be useful in taking students mental model development into consideration when developing future curricula for introductory programming courses. We also encourage further investigation of the use of Bebras puzzles as a research tool in order to develop a structured and validated approach to deploying them in computer science education research. The initial results presented here are promising for their use.

ACKNOWLEDGMENT

The authors would like to thank the Year in Computing students of 2017 for their willingness to participate in this study.

REFERENCES

[1] D.N. Rapp, "Mental models: theoretical issues for visualizations in science education", in Visualization in Science Education, J.K. Gilbert, Editor. Springer Netherlands: Dordrecht. pp. 43-60, 2005.

[2] V. Ramalingam, D. LaBelle and S. Wiedenbeck, "Self-efficacy and mental models in learning to program", in Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. ACM: Leeds, United Kingdom. pp. 171-175 2004.

[3] M.E. Caspersen, K.D. Larsen and J. Bennedsen, "Mental models and programming aptitude", in Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. 2007, ACM: Dundee, Scotland. pp. 206-210, 2007.

[4] R. Bornat, S. Dehnad, and Simon, "Mental models, consistency and programming aptitude", in Proceedings of the tenth conference on Australasian computing education - Volume 78, Australian Computer Society, Inc.: Wollongong, NSW, Australia. pp. 53-61, 2008.

[5] S. Dehnadi, "Testing programming aptitude", in P. Romero, J. Good, E. A. Chaparro & S. Bryant, eds, 'Proceedings of the PPIG 18th Annual Workshop', pp. 22–37, 2006.

[6] Bebras. Bebras - "International challenge on informatics and computational thinking". 2003 [cited 2017 1st. of October]; Available from: http://www.bebras.org/.

[7] P.N. Stern, "Grounded theory methodology: Its uses and processes". Image. 12(1): pp. 20-23, 1980.

[8] A. Strauss and J. Corbin, "Grounded theory methodology. Handbook of qualitative research". 17: pp. 273-85, 1994.

[9] D.L. Rennie, "Grounded theory methodology as methodical hermeneutics: reconciling realism and relativism", in Theory & Psychology. 10(4): pp. 481-502, 1994.

[10] A. Bryant and K. Charmaz, "The Sage handbook of grounded theory". Sage. 2007.

[11] J. Sorva, "Notional machines and introductory programming education", in Trans. Comput. Educ., 13(2): pp. 1-31, 2007.

[12] B. Du Boulay, T. O'Shea and J. Monk, "The black box inside the glass Box: Presenting computing concepts to novices", in Int. J. Man-Machine Studies, 14, pp. 237-249, 1981.

[13] J. Wai, D. Lubinski, and C.P. Benbow, "Spatial ability for STEM domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance", in Journal of Educational Psychology, 101(4): p. 817, 2009.

[14] R.B. Miller, G.N. Kelly, and J.T. Kelly, "Effects of Logo computer programming experience on problem solving and spatial relations ability", in Contemporary Educational Psychology, 13(4): pp. 348-357, 1988.

[15] S.J. Jones and G.E. Burnett, "Spatial skills and navigation of source code", in Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education. ACM: Dundee, Scotland. pp. 231-235, 2007.

[16] S. Jones and G. Burnett, "Spatial ability and learning to program", in Human Technology: An Interdisciplinary Journal on Humans in ICT Environments, 2008.

[17] S. Cooper et al. "Spatial skills training in introductory computing", in Proceedings of the eleventh annual International Conference on International Computing Education Research. ACM 2015.

[18] A. Solomon, "The role of spatial reasoning in learning computer science", in Proceedings of the 2016 ACM Conference on International Computing Education Research. ACM: Melbourne, VIC, Australia. pp. 265-266, 2016.

[19] M. de Raadt et al., "Approaches to learning in computer programming students and their effect on success", in Research and Development in Higher Education Series, 2005.

[20] S. Fincher et al., "Programmed to succeed?: A multi-national, multi-institutional study of introductory programming courses", The Computing Laboratory, University of Kent, 2005.

[21] Simon et al., "Predictors of success in a first programming course", in Proceedings of the 8th Australasian Conference on Computing Education - Volume 52, Australian Computer Society, Inc.: Hobart, Australia. p. 189-196, 2006.

[22] R.B. Ekstrom et al., "Manual for kit of factor-referenced cognitive tests", Princeton, NJ: Educational testing service, 1976.

[23] D. Reilly, D. Neumann and G. Andrews, "Gender differences in spatial ability: Implications for STEM education and approaches to reducing the gender gap for parents and educators", in M. S. Khine (Ed.), Visual-Spatial Ability: Transforming Research into Practice pp. 195-224, 2016.

[24] Y. B. Kafai, "Chapter 4: Video game designs by girls and boys: Variability and consistency of gender differences", MIT press, 1998.

[25] Bebras-UK, "2013 Self-marking questions (Senior) - Spinning toy", 2013 [cited 2017 1st of October]; Available from: https://challenge.bebras.uk/index.php?action=user_competitions.

[26] Bebras-UK. "Practice hallenge 2015 (Elite (age 16-18)) - Drawing starts", 2015 [cited 2017 1st of October]; Available from: https://challenge.bebras.uk/index.php?action=user_competitions.

[27] Y. Byung-Do, and O.N. Garcia. "Cognitive activities and support in debugging", in Proceedings Fourth Annual Symposium on Human Interaction with Complex Systems, 1998.