Students' behavioral intention to use gradual programming language Hedy: a technology acceptance model

Yeni, S.; Meulen, A.N. van der

Students' Behavioral Intention to Use Gradual Programming Language Hedy: A Technology Acceptance Model

Sabiha Yeni
Leiden University, LIACS
The Netherlands
s.yeni@liacs.leidenuniv.nl

Anna van der Meulen
Leiden University, LIACS
The Netherlands
a.n.van.der.meulen@liacs.leidenuniv.nl

ABSTRACT

Programming is generally considered a cognitively challenging subject by beginners because it involves acquiring complex new knowledge, strategies, and practical skills. Effective instructional strategies and programming platforms are important in providing the student with optimal learner support. As a new approach, the first gradual programming language Hedy was launched to overcome issues with syntax and cognitive overload when learning to program by teaching syntax and semantic knowledge in steps, rather than at once. In this paper, we aim to investigate students' acceptance of Hedy, by analyzing their behavioral intentions from the theoretical background of the Technology Acceptance Model (TAM). We conducted a qualitative case study combining a survey and group interviews to capture the dimensions of TAM. 18 students between the ages of 10 and 12 attended four days of summer camp on Hedy during which data were gathered. The results indicate that several of the participants have a positive attitude towards Hedy, and their experience of "not too difficult but real programming" appears in line with the intention of the gradual programming language. However, some other children found Hedy too limited and restricted or expressed a desire for a different type of output. Overall, learners' experience appeared embedded in their previous programming experience as well as expectation beforehand of Hedy and of programming in general. Finally, a trend could be seen where learners from under-resourced communities (about half of the participants) overall were less positive about their experience and behavioral intention.

CCS CONCEPTS

• **Applied computing** → *Education*; • **Social and professional topics** → **K-12 education**; *Computer science education.*

KEYWORDS

gradual programming language, Hedy, the technology acceptance model

ACM Reference Format:
Sabiha Yeni and Anna van der Meulen. 2022. Students' Behavioral Intention to Use Gradual Programming Language Hedy: A Technology Acceptance Model. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3502718.3524782

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland.

ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524782

1 INTRODUCTION

In recent years, the demand for programming education and student interest in programming have grown rapidly, and introductory programming courses have gained increasing popularity. However, programming courses are generally regarded as difficult, and computer science programs have dropout rates up to 40% [3], which is high compared to other programs traditionally considered difficult such as physics. This high dropout rate has been attributed to existing instructional strategies [3] and unrealistic expectations [21]. This might be due to the fact that learning to program involves acquiring complex new knowledge, strategies and practical skills [31]. Regarding the practical skills of programming, literature has identified problems with understanding and mastering the programming syntax and functions, applying correct syntax rules, using semantic knowledge to write programs and ineffective design and testing solutions [16].

To address issues with syntax when learning to program, the idea of a gradual programming language was recently coined. A gradual language teaches syntax in steps, rather than at once [14]. As is common in both mathematics and natural language teaching, learners using a gradual language initially learn incomplete and partly incorrect models, which are refined step by step. The programming language Hedy is an implementation of this concept. Hedy is an open-source programming language that runs in the browser and is available for free. Previous research on Hedy has manually examined almost 10.000 Hedy programs to gain an understanding of how novices learn Hedy [14], and have analyzed users' feedback to understand benefits and challenges [10]. In our empirical study, we focus on students' acceptance of a gradual programming language, by analyzing their behavioral intentions towards the use of Hedy. We also look explicitly at the role of students' socioeconomic background (SES) by involving a group of learners from under-resourced communities. Our research question is: **What are the behavioral intentions of students toward the use of gradual programming language, and are there different patterns for students from under-resourced communities?**

2 BACKGROUND

2.1 Cognitive Load and Syntax Issues

Programming demands complex cognitive skills such as procedural, conditional and analogical reasoning, and planning [18]. Lister argued that cognitive *over*load can occur in students while programming because of simultaneous teaching of programming concepts, syntax and problem solving [20]. A worldwide study in 250.000

novice programmers identified mismatched brackets and quotations as the most common error category in writing syntax, a fundamental part of programming [1]. It has been suggested that learning the style and structure of programs are preceded by learning the syntax [24]. Once students have internalized syntax, they should be able to ignore the details of the syntax during program design and construction, allowing them to direct attention to more relevant parts of the program [30].

Because of the cognitively challenging nature of programming, good instructional strategies are important to provide the student with optimal learner support. Mason et. al [23] documented that some of the programming environments may be complex and cause an adverse impact on learners' focus of attention and overloading cognitive resources for learning [23]. Consequently, the idea of designing easy-to-use programming platforms may be taken into consideration [26]. A gradual programming language [14] [10] forms a new approach in simplifying programming for novices. It presents a new way of teaching the increasingly complex syntax of programming language, based on how punctuation is taught to novice readers in natural language education. The syntax rules are introduced slowly and gradually while students learn to write their first natural language.

2.2 Gradual Programming Language: Hedy

Hedy is the first implementation of gradual programming. The user interface of Hedy is shown in Figure 1. On the left is an editor for entering code, and on the right is a field for output. Each level also includes buttons that allow you to practice the instructions you've learned thus far. Hedy introduces syntax and programming concepts in an incrementally increasing manner with the levels.

In addition to the gradual approach, Hedy has other features that make learning programming easier. The first is the palette of available commands. The user interface contains an overview of the commands available in a level, integrated into the editor, as shown in Figure 1. This saves the users the cognitive effort of looking up commands. Each of the commands has a "Try this" button, which will place a code snippet containing that command in the editor. Second, there are the built-in instructions and exercises for each level. An integrated tutorial is enabled by the gradual programming paradigm since the small size of each level contributes to brief explanations in a clear order.

Finally, we give a brief overview of the first eight levels of Hedy taught in this study (more detailed descriptions of all 18 levels can be found in previous papers [14] [10]). At level 1, students can print text with no other syntactic elements than the keyword `[print]` followed by arbitrary text (Figure 1). In the next consecutive levels, students add variables (using the word `[is]` rather than the equals symbol) (level 2) and create lists and retrieve elements (level 3). Next, at level 4 the first syntactic element is introduced: the use of quotation marks to distinguish between variables and "plain text", and selection with the `[if]` statement is introduced (level 5). Students learn to calculate when addition, multiplication, subtraction and division are introduced in level 6, and in level 7 repetition with a simple syntactic construction is introduced. Finally, in level 8, learners will see a more Python-like form of the loop, namely: `[for i in range 0 to 5]`. (In the most recent version of Hedy, "for" syntax is introduced in Level 10).

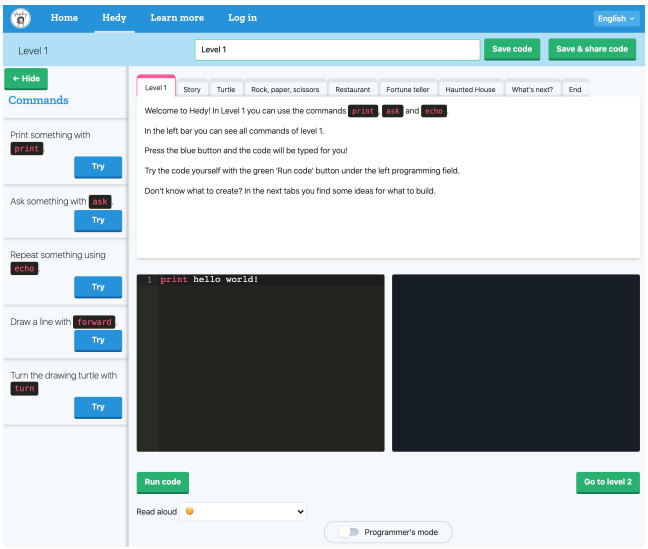

Figure 1: Level 1 of the Hedy user interface in English

2.3 The Technology Acceptance Model

The technology acceptance model (TAM) [7] [8] is one of the most widely used theoretical models to understand and predict user acceptance of technology (Figure 2) [37] [6] [15]. TAM is predictive in nature and aims to find the factors that influence people's intentions to use technology.

TAM comprises core variables of user motivation: perceived ease of use (PEU), perceived usefulness (PU) and attitudes toward technology, as well as outcome variables: behavioral intention and technology use [22]. PEU is a degree of ease of using particular technology and learning without additional effort [8]. If people think that new technology is easy to use, their behavioral intention towards using technology becomes positive. PU refers to a positive or negative idea on performance increase in users' jobs as emerged after using technology [8]. Further, behavioral intention (BI) can be explained as people's presence to act a specified behavior. TAM states that behavioral intention of people is the primary factor that determines people's actual use [34]. The purpose of this study to apply the original TAM model [8] as a baseline model to examine the students' acceptance of a gradual programming language.

2.4 The Role of Socio-Economic Status in Computing

Regarding the attitude towards technology use and computing, socioeconomic status (SES) is known to be an important factor [27]. Generally, students with lower SES can be impacted by less concrete access to technology and computers as well as several other social factors that decrease their familiarity and proficiency in this area [11]. Consequently, in the context of understanding and improving young learners' behavioral intention towards programming it is important to pay attention to this group.

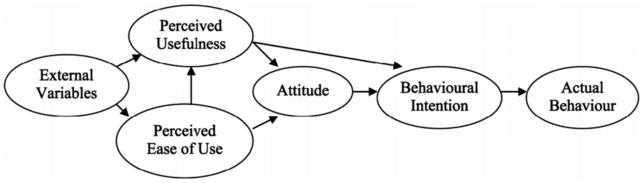

Figure 2: Technology acceptance model [8]. ***(External variables in this study are classified into individual factors: prior experience and programming self-efficacy)***

Because a core element of the gradual programming approach is the aim to simplify programming for novice learners, TAM is a very suited approach to explore how this affects students' perceived ease of use, user motivation and overall attitude. In this study, the gradual programming language Hedy was introduced to upper primary school students (including students from under-resourced communities) via a four-day summer camp. Then, we used TAM to evaluate the behavioral intentions of students toward the use of Hedy.

3 METHODOLOGY

This study employed a qualitative case study approach [32] to explore students' behavioral intention toward the use of a gradual programming language. Within this approach, the language Hedy is our specific unit of interest [28].

3.1 Participants

In total, 18 students between 10-12 years old (average age 11) participated in the study. The participants were three girls and 15 boys from 9 different schools in the Netherlands. Eight students were considered to come from under-resourced communities, based on their eligibility to enter the camp through a free ticket, as enabled through their schools (referred to below as group with "low-SES"). In providing their demographics, students also indicated their previous programming experience. Eight students took a programming course before, only one participated in a course longer than one month. Six students did not take a course but tried to learn programming by themselves, and three students did not take a course or try to learn themselves (one student did not fill out the question). We also asked students how comfortable they feel when they are programming. Seven students stated that they feel as programming professionals, five students feel comfortable while programming, five students need a little help and one student a lot of help.

3.2 Hedy Lessons

The study took place within the context of a summer camp on Hedy, conducted at Leiden University. All participants of the summer camp also took part in the study. The camp lasted four days, between 9 am and 4 pm, covering the first 8 levels of Hedy. Each day of lessons had a similar setup. First, the teacher gave instructions by explaining the basic concepts of a level. Next, the students could pick their preference for an assignment out of several provided options. The students were allowed to work independently but were encouraged to show their programs to other students at the end of each level. Questions could be asked when needed. During the camp days several other CS unplugged activities and games related to programming were conducted with the participants, including "Play a robot", code-card games and board games about programming. Further, Hedy lessons and programming activities were interchanged through the day with breaks where the students could play computer games or have free time outside.

The camp was guided by a researcher and a certified primary school teacher with experience in teaching programming and Hedy, who were assisted by three bachelor's and master's computer science students. The authors of this study were independent observers and not part of the camp team.

3.3 Data Collection

The data were gathered using a demographic survey, Technology of Acceptance (TAM) model survey and group interviews. The demographic survey was completed at the start of the camp, and included questions on age, gender, grade, programming experience and self-efficacy. The topics of the survey and group interview were closely aligned, with the purpose to obtain a broad, quantitative overview through the survey complemented with in-depth experiences and opinions obtained through the group interviews. We choose group interviews since for younger participants this is a more natural and comfortable situation to share.

The TAM survey included 22 questions and was adapted from an existing survey [19] (based on the original TAM model [7]) to fit the purpose of our study. Previously [19], survey reliability and validity as assessed via confirmatory factor analysis were both found acceptable [9]. Our adapted version of the survey included questions on prior experience, self-efficacy, attitude towards Hedy and camp experience, perceived ease of use, perceived usefulness and behavioral intention (survey available upon request). The "Degree of happiness scale", a version a 5 point Likert scale suited for children, was used to indicate the level of agreement, ranging from (5) Strongly agree to (1) Neutral, (in line with young users tendency to rate relatively positively) with icons of smiley's to illustrate the ratings [12]. The group interviews were conducted following the end of lessons on the fourth day. All interviews were audio recorded. We conducted five interviews. Group 1 to 4 had four participants and group 5 had two participants. Division of low-SES students was as follows: group 1, 2, and 3 had two out of four students with low-SES, in group 4 there were no low-SES students and in group 5 both students had low-SES. The interview followed a semi-structured interview protocol that included six subgroups to capture students' intention towards Hedy: **prior experience, self-efficacy, attitude towards Hedy and camp, perceived ease of use, perceived usefulness, behavioral intention.**

3.4 Data Analyses

The survey data were analyzed using the Statistical Package for the Social Sciences (SPSS), version 27. Sum and average scores as well as mean, SD, range, and internal consistency (Cronbach's alpha) were calculated for each subgroup. The role of low-SES was explored through a comparison for each subgroup between low-SES and other children.

The group interviews were transcribed using a standard verbatim approach, noting also behaviors such as laughter or sighing. Pseudonyms (Child1 etc.) were used in the transcript, and personal pronouns have been converted to a singular gender-neutral pronoun (they/them/their). Next, a theory driven thematic data analysis approach was used [5]. The themes were in line with subgroups of the TAM survey: behavioral intention, perceived ease of use, perceived usefulness, external variables (self-efficacy and prior experience) and attitude. In the final step, the quantitative and qualitative data were taken together per subgroups, as described below in the results. Quotes from the interviews were added to illustrate certain points where applicable.

4 RESULTS

4.1 Prior experience with programming in general

The survey inquired with two questions about students' prior experience with programming, asking whether they previously enjoyed and felt comfortable while programming. Scores received on this subgroup of the survey ranged from 1 to 5, with an average score of M = 3.72 (SD = 1.27). No differences between the low-SES (M = 3.75, SD = 1.51) andother children (M = 3.70, SD = 1.13) was observed.

During the interview, the students talked about programming tools or languages they previously used. In several groups all children indicate to have prior programming experience (groups 1, 4, and 5), mentioning for instance Scratch, Micro:bit, and Cospaces [https://cospaces.io/]. The children were not very specific about their experience which appears sometimes limited. Often the children directly make comparisons to Hedy, which are incorporated in the next section.

4.2 Programming Self-Efficacy

Three items formed the subgroup "self-efficacy" about programming and Hedy. Scores ranged from 1 to 4.67, M = 3.09, SD = 1.17. Internal consistency was good, α = .80. Low-SES learners scored lower (M = 2.54, SD = 1.22) compared to the other learners (M = 3.53, SD = .97).

In the interviews the children rated themselves between 4 and 8 (on a scale of 10) in terms of how skilled programmers they are. In two groups (2 and 3) the amount of help from the teachers needed is mentioned as an indication of being not so skilled yet, as well as (in group 2) that it is difficult when something new arises. Child 1 in group 5 is most clearly negative, indicating they "understand nothing from computers". Further, it is discussed what happens when the children encounter a mistake. The most common indicated strategy is to ask for help, but more than half of children also mention either to first try themselves (carefully checking the code, starting over again, taking a break from it) or to imagine being able to solve by themselves when no one is around. Concerning the latter, several children (Child 2 in group 5, several children in group 4 and group 2) seem pretty confident that they can work it out themselves, whereas especially Child 3 (group 2) is certain they could not since they are only at level 7.

4.3 Attitude towards Hedy and camp experience

While the survey focused mostly on participants' experience with the camp, in the interviews the children talked elaborately both about the camp and their overall experience of Hedy. First, concerning the summer camp, ratings (based on 7 questions) ranged from 1.14 to 5 with an average score of M = 4.17 (SD = .98). Internal consistency of 7 item subgroup was good, α = .94. A trend was observed with low-SES students scoring lower (M = 3.71, SD = 1.29) compared to the other students (M = 4.54, SD = .43).

Second, the interview questions on this topic showed that most participants are enthusiastic; finding their experience with programming during the camp fun, interesting, educational. Three children bring up that they see working with Hedy as "actual coding" (Child 4, group 4), compared to their previous experience which "wasn't really programming", "not really like they do it here" (Child 2, group 5). In group 4 one child also identifies Hedy as a bit of a mix between Scratch and Python since "it is not so hard with all those punctuation and all but it is just writing". There is however also some criticism and hesitation. First of all, there are specific children who have an overall less positive attitude. This includes the case of Child 1 (low-SES student) in group 5, who's experience is strongly impacted by the fact that they find the days of the camp too long. In group 3 as well three children (all except Child 2) are less positive. For them (two of whom are low-SES students) it appears a mix of having a different expectation, encountering errors/bugs in the program, as well as being bothered by the cost of the camp. Second, overlapping with some of these points, there are themes that come up within the overall group of children, including those who are generally positive. Across the groups the children mention having had different expectations before the camp. Not all of them can explain why, but six children (in groups 1, 2, 3 and 5) indicate that they would prefer or expected other output, creating something tangible or a game. Also, for five children it is a problem that Hedy still has bugs, one child however states that it was cool to find the bugs, because it made them feel smarter than those who made the program.

4.4 Perceived Ease of Use

Three items in the survey formed the subgroup "perceived ease of use". Scores ranged between 1.67 and 4.33, M = 3.47 and SD = .82. Internal consistency was just below acceptable, α =.57. Especially the item "I find learning programming with Hedy easy" appeared to stand out. Low-SES students (M = 3.0, SD = .76) tended to score lower compared to the other students (M = 3.90, SD = .65).

In the interviews, 9 children from all groups except group 3 found Hedy to be pretty easy. They stated "...because you can start working directly, and because the explanations, examples, and try button are helpful". In group 5 Child 2 indicates: "I think it is really a language that is easy to learn for children [...] if you make a mistake for example, a pretty big mistake, then Hedy really shows you exactly what the mistake is". Some children are more hesitant, expressing that it is difficult because of the change and increasing difficulty between levels, or that switching between English and Dutch is hard. Finally, Child 1 of group 5 generally finds programming and computers difficult and panic inducing.

4.5 Perceived Usefulness

Two items in the survey formed the subgroup "perceived usefulness". Scores ranged between 1 and 5, M = 3.64, SD = 1.52. Internal consistency of the subgroup was good, α = .88. Low-SES students scored lower (M = 3.19, SD = 1.87) compared to other students (M = 4.00, SD = 1.18).

The interviews showed that most children agree it is useful to learn programming in general (with the exception of Child 1 in group 5 being most hesitant about this for themselves personally) but they have some nuanced ideas on the usefulness of Hedy. While 8 children are positive here as well and mention specific benefits (including that it is educational because of examples given on commands, and that it can help in between Scratch and Python which are quite different), a theme can be seen concerning the generalized usefulness of what is being learned. This is expressed clearly in group 1 (Child 1 and Child 2) and group 3 (Child 2) as well as by some other children. The former children seem to have nuanced opinions about this, expressing both that programming should be open and not so pre-determined and that it should focus directly on a language you will use later on. At the background here there is also an idea it is especially beneficial for (some) children starting with programming. Child 2 (group 4): “yes it is for the start, but it still bothers me after that there is nothing you can do with it”.

4.6 Behavioral Intention

Behavioral intention was based on three items. Scores on this subgroup ranged from 1 to 5, M = 3.31, SD = 1.52. Internal consistency was good, α = .83. Low-SES students tended to score lower (M = 2.83, SD = 1.74) compared to the other students (M = 3.68, SD = 1.29).

During the interviews, in each group there were children who imagined continuing with Hedy later on, though not everyone was certain or indicated a preference. Some specifically mention that they want to finish the levels, some have concrete ideas on making something for school or at some point being able to make games. Most convinced they will not continue are Child1 (group 1) because they think it is boring and they prefer Tumble, and Child1 (group 5) who just finds it really difficult. A later inventory in the Hedy database showed that five children did in fact later continue. Three of these students had expressed this during the interview, whereas one (low-SES student) had been hesitant and the other one was the child who thought Hedy to be boring.

5 CONCLUSION AND DISCUSSION

In this study, we explored the intention of students to use a gradual programming language Hedy based on TAM, as well as the patterns for students from under-resourced communities. 18 students aged 10-12 followed four days of summer camp on Hedy, after which data were collected through the TAM survey and group interviews.

Our results show that specific characteristics of gradual programming appear to positively affect almost all learners’ evaluation. The children appreciate that it is “easy to learn” yet at the same time “real programming”. Also, the specific educational features are brought forward by the children themselves as helpful. They find the examples given on commands educational and place it in between Scratch and Python which are quite different. Similar findings were reported in the study [10], where students appreciated the gradual nature of Hedy, find Hedy easy to learn and especially like and use the built-in education features such as example code snippets [10].

Further, there is a difference in motivation between students related to code outputs. It can be seen that especially for some children text-based output is a disadvantage. It could be that for some, the motivation comes from the text-based programming (making gradual Hedy very suited, since easy but real!) but for others it comes from the visual or tangible outputs, and for the latter group Hedy is less motivational. This might be related to students’ previous experiences with visual programming languages or robotics which are popular for introducing programming to children quickly. Lack of syntax and outputs of languages make these languages more attractive for some of them [17]. However, research also shows that it merely delays syntax problems or even makes students believe that syntax errors are not that important causing them to create codes that can not be compiled [29]. A related issue is that the majority of students had difficulties while transferring block-based programming knowledge (Snap!) to text-based language (Java) [35]. As parallel to our findings, some students (especially high school and university students) are concerned that block-based programming languages are not “real” compared to text-based programming languages [35], although they use a variety of programming structures and concepts [33]. Another option might be hybrid programming languages (e.g. hybrid form of Java) to bridge between block-based and text-based programming [25]. Hybrid programming languages could also be used for students struggling with the text-based approach or additionally to provide a challenge to younger learners who are bored with the visual programming approach.

Continuing on this point, there appear to be different “user profiles", embedded in factors such as previous experience, expectations specifically of Hedy and of programming in general. In addition to the children mentioned above for whom a certain type of output appears most central in their motivation, within the evaluation of the text-based aspects there are children who, although generally not very negative about Hedy, show clear and pretty nuanced hesitations within their perceived usefulness and behavioral intentions. It seems they expect and want more in terms of freedom and challenge of programming and see value in other languages. On the other hand, there are children for whom Hedy is in fact already “more real or actual programming". Consequently, Hedy might fit some learners better than others, yet this could possibly be addressed by paying attention to new learners’ perceptions and expectations (for instance, emphasizing that the end of Hedy really is the beginning of Python). Expectancy-Value Theory, which posits that students’ motivations, achievements, and choices are determined by their expectations and their subjective task values related to specific activities, can explain this situation [36].

Finally, there was a clear indication for the role of socio-economic status. The quantitative data show a consistent trend with students from under-resourced communities being less positive toward Hedy, finding it more difficult and less useful, and having lower intention to continue. Although there is some back-up of this in the qualitative data (where three of the overall clearly most negative kids are low-SES learners), it also seems the low-SES students are in general less outspoken in the interviews. Moreover, there are in our group also low-SES students who have a different profile (for

instance Child 2 in group 5 who is quite positive). Previous research shows that students from low-SES are expected to have less access to technology outside of the classroom compared to their higher SES peers [4]. Further, other research indicates that unequal material access and mental access (focuses on the psychological and emotional burdens placed upon those who lack experience with technology, such as computer anxiety) leads to different usage patterns between children when using technology [2]. Prior research has revealed that students' low-SES had the largest negative direct effect on attitudes toward technology use [4]. Adding our findings, it would be valuable to attempt to further understand how low-SES students can be motivated and included. It could be that especially for this group addressing their perceptions and expectations can be beneficial.

Concluding, using the TAM model we gained insight in "ease of use" and "usefulness" as central factors that affects learners' experience and behavioral intention with the gradual programming language Hedy. Future studies could within a larger group quantitatively map out the interrelations between the different factors, and further assess the role of factors such experience and general perception of programming. The latter could also be explored in order to understand whether attention for perceptions and expectations can help in particular to motivate and include low-SES students.

6 LIMITATIONS

This study has some limitations. First, for the interviews there was quite some variance between the groups in the order in which and how elaborately each topic was discussed, related to responsiveness of the children and atmosphere (with especially group 3 being a very active and restless group). This is however in line with the set-up of a semi-structured interview as well as with the recommendations of how to accommodate research with children [13]. Second, the summer camp was limited to only four days, and it is estimated that if children had spent more time with Hedy and gained more knowledge about other levels of Hedy, it might have influenced their intention to use Hedy.

REFERENCES

[1] Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In *Proceedings of the ACM SIGCSE '15* (Missouri, USA). 522–527.

[2] Christopher Ball, Kuo-Ting Huang, RV Rikard, and Shelia R Cotten. 2019. The emotional costs of computers: An expectancy-value theory analysis of predominantly low-socioeconomic status minority students' STEM attitudes. *Information, Communication & Society* 22, 1 (2019), 105–128.

[3] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin* 37, 2 (2005), 103–106.

[4] Courtney K Blackwell, Alexis R Lauricella, and Ellen Wartella. 2014. Factors influencing digital technology use in early childhood education. *Computers & Education* 77 (2014), 82–90.

[5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[6] Meyrick Chow, David Kurt Herold, Tat-Ming Choo, and Kitty Chan. 2012. Extending the technology acceptance model to explore the intention to use Second Life for enhancing healthcare education. *Computers & education* 59, 4 (2012), 1136–1144.

[7] Fred D Davis. 1986. *A technology acceptance model for empirically testing new end-user information systems: Theory and results.* Ph. D. Dissertation. MIT.

[8] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.

[9] Claes Fornell and David F Larcker. 1981. Evaluating structural equation models with unobservable variables and measurement error. *Journal of marketing research* 18, 1 (1981), 39–50.

[10] Marleen Gilsing and Felienne Hermans. 2021. Gradual Programming in Hedy: A First User Study. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VL/HCC51201.2021.9576236

[11] Joanna Goode. 2010. Mind the gap: The digital dimension of college access. *The Journal of Higher Education* 81, 5 (2010), 583–618.

[12] Lynne Hall, Colette Hume, and Sarah Tazzyman. 2016. Five degrees of happiness: Effective smiley face likert scales for evaluating with children. In *Proceedings of the the 15th international conference on interaction design and children*. 311–321.

[13] Libby Hanna, Kirsten Risden, and Kirsten Alexander. 1997. Guidelines for usability testing with children. *interactions* 4, 5 (1997), 9–14.

[14] Felienne Hermans. 2020. Hedy: A Gradual Language for Programming Education. In *Proceedings of the 2020 ACM Conference on ICER*. 259–270.

[15] Richard J Holden and Ben-Tzion Karsh. 2010. The technology acceptance model: its past and its future in health care. *Journal of biomedical informatics* 43, 1 (2010), 159–172.

[16] Mohd Ismail, Nor Ngah, and Irfan Umar. 2010. Instructional strategy in the teaching of computer programming: A need assessment analyses. *The Turkish Online Journal of Educational Technology* 9 (04 2010).

[17] Divna Krpan, Saša Mladenović, and Goran Zaharija. 2017. Mediated transfer from visual to high-level programming language. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 800–805.

[18] D Midian Kurland, Roy D Pea, Catherine Clement, and Ronald Mawby. 1986. A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research* 2, 4 (1986), 429–458.

[19] Yi-Hsuan Lee, Yi-Chuan Hsieh, and Yen-Hsun Chen. 2013. An investigation of employees' use of e-learning systems: applying the technology acceptance model. *Behaviour & Information Technology* 32, 2 (2013), 173–189.

[20] Raymond Lister. 2016. Toward a developmental epistemology of computer programming. In *Proceedings of the 11th workshop in primary and secondary computing education*. 5–16.

[21] Andrew Luxton-Reilly. 2016. Learning to program is easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 284–289.

[22] Nikola Marangunić and Andrina Granić. 2015. Technology acceptance model: a literature review from 1986 to 2013. *Universal access in the information society* 14, 1 (2015), 81–95.

[23] Raina Mason, Graham Cooper, B Simon, and Barry Wilks. 2015. Using Cognitive Load Theory to select an Environment for Teaching Mobile Apps Development.. In *ACE*. 47–56.

[24] Evelyn Ng and Carl Bereiter. 1991. Three levels of goal orientation in learning. *Journal of the Learning Sciences* 1, 3-4 (1991), 243–271.

[25] Mark Noone, Aidan Mooney, and Keith Nolan. 2020. Hybrid Java: The creation of a hybrid programming environment. *Irish Journal of Technology Enhanced Learning* 5, 1 (2020).

[26] Sharon Oviatt. 2006. Human-centered design meets cognitive load theory: designing interfaces that help people think. In *Proceedings of the 14th ACM international conference on Multimedia*. 871–880.

[27] Miranda C Parker, Amber Solomon, Brianna Pritchett, David A Illingworth, Lauren E Marguilieux, and Mark Guzdial. 2018. Socioeconomic status and computer science achievement: Spatial ability as a mediating variable in a novel model of understanding. In *Proceedings of the 2018 ACM Conference on ICER*. 97–105.

[28] Michael Quinn Patton. 2014. *Qualitative research & evaluation methods: Integrating theory and practice.* Sage publications.

[29] Kris Powers, Stacey Ecott, and Leanne M Hirshfield. 2007. Through the looking glass: teaching CS0 with Alice. In *Proceedings of the 38th SIGCSE*. 213–217.

[30] Robert S Rist. 1989. Schema creation in programming. *Cognitive Science* 13, 3 (1989), 389–414.

[31] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[32] Robert E Stake. 1994. Case study: Composition and performance. *Bulletin of the Council for Research in Music Education* (1994), 31–44.

[33] Leigh Ann Sudol. 2009. *Visual Programming Pedagogies and Integrating Current Visual Programming Language Features.* Ph. D. Dissertation. Robotics Institute.

[34] Timothy Teo. 2011. Factors influencing teachers' intention to use technology: Model development and test. *Computers & Education* 57, 4 (2011), 2432–2440.

[35] David Weintrop. 2015. Minding the gap between blocks-based and text-based programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. 720–720.

[36] Allan Wigfield and Jenna Cambria. 2010. Expectancy-value theory: Retrospective and prospective. In *The decade ahead: Theoretical perspectives on motivation and achievement*. Emerald Group Publishing Limited.

[37] Sabiha Yeni and Zeynep Gecu-Parmaksiz. 2016. Pre-Service Special Education Teachers Acceptance and Use of ICT: A Structural Equation Model. *Journal of Education and Training Studies* 4, 12 (2016), 118–125.