Exploring Creativity Expectation in CS1 Students' View of Programming

Ville Isomöttönen Antti-Jussi Lakanen Paavo Nieminen
Faculty of Information technology
University of Jyväskylä
Jyväskylä, Finland
ville.isomottonen@jyu.fi, antti-jussi.lakanen@jyu.fi, paavo.nieminen@jyu.fi

***Abstract*—Full paper in Research category: Literature provides creativity definitions that are applicable to educational settings. For example, the definition by Plucker et al. emphasizes the 'social context' in which the usefulness and novelty of a creative outcome is evaluated, and notes that this emphasis allows students' coursework to be deemed creative without extraordinary characteristics. Computing educators tend to assume that incoming CS course populations welcome creativity, and utilize application contexts (e.g., games, media, arts, and robots) in which creativity is a central attribute. Previous research also suggests that beginner CS students may initially possess versatile identities regarding what computing will entail. This article seeks to provide further evidence for creativity expectation among students starting a CS1 course, looking at how and to what extent creativity is acknowledged. This agenda was observed to be possible using a large data set (N=1,946, eight-year period) in which students at the very beginning of their computing studies characterized what programming is. Qualitatively different creativity-related categories were identified and frequencies for these categories were counted in a sample of 240 respondents. Further content analysis was applied to the remaining data by using word searches. The categories identified were: freedom to create and express, creativity needed in problem solving, programming as a circumstance for personalized activity, associations with arts, creative innovations, tolerance to open-ended situations, aesthetics and elegance of programming, and programming as a flow-like activity. In the sample of 240, 59% of the data was interpreted to refer to creativity, and among the word-searched portion of the data, nearly one-third was interpreted to indicate it. The illustrations and the numbers of references support educators' assumptions as they consider introducing creativity-related education.**

***Index Terms*—introductory programming, creativity**

I. Introduction

This article reports on beginner students' view of programming by focusing on how they referred to creativity. Motivation is provided by at least two features in the computing education literature. First, the work by Peters [1] revealed that students entering computing programs may have versatile views of what computing is with respect to their developing identity. Her observation was that these initial identities may become less versatile during the course of study. By documenting creativity expectations among beginners in a large data set, the present study explores if support is found for the observation of students' initial versatile views of computing.

Second, motivation is provided by the literature that reports positive effects of the use of application contexts in CS education. In a study by Greenberg et al. [2], students who were offered a programming course in a contextualized form showed increased intentions to take further CS courses. Similarly, Guzdial [3] summarized several-year experiences of media computing contextualization by emphasizing the importance of application context. He reported that students have welcomed the opportunity to rely on their creativity in this setting. The present study seeks to extend evidence in this respect.

As potential support for the lines of research above, and for the educators' anecdotal assumptions such as "Most people enter the field of computer science with a desire to create something new" [4], creativity-related considerations among beginner students were explored. A data set initially collected with a general interest in how beginners characterize programming was observed to make this agenda possible. This data results from a questionnaire that has been annually issued to students at the very beginning of their introductory programming course (CS1). A total of 240 longest data entries out of 1,946, collected during eight years, were manually examined and peer-checked using qualitative content analysis. In order to yield a complete overview of the creativity aspect, the remaining data was further explored using word searches.

Section II reports definitions of creativity and the relevant related work in CSEd literature. Section III describes the study context and the method. Section IV describes the result categories and their frequencies. Conclusions and proposals for future work are stated in Section V.

II. Background

A. Creativity definitions

From the vast literature available on education and creativity, this section presents a few definitions that were deemed relevant to the present study.

In Sternberg's [5] definitions, creativity comprised three abilities, which were *synthetic*, *analytic*, and *practical*. The synthetic refers to an ability to generate new ideas. The analytic refers to an ability to analyze and evaluate ideas. The practical refers to an ability to put ideas into practice. These perspectives alone seem to aptly characterize programming as creative problem solving. Sternberg also emphasized self-efficacy as a factor affecting how it is possible to engage students in activities that require creativity.

Plucker et al. [6] proposed the following definition:

> "Creativity is the interaction among aptitude, process, and environment by which an individual or group produces a perceptible product that is both novel and useful as defined within a social context."

This definition was informed by a review of literature regarding how creativity was referred to. Plucker et al. observed that the literature lacked a common definition, and that many studies did not include any definition or only implicitly described creativity. Moreover, they reported typical myths about creativity, of which one important aspect for educators was 'Big C,' that is, the conception that creativity belongs to exceptional individuals. Their definition above included 'social context' to emphasize that creativity can be looked at in a particular setting and student ideas can be valued without the worry of Big C. Moreover, the authors explained that they preferred to use the term 'aptitude' in place of 'trait,' not to refer to innate characteristics that cannot be practiced. 'Perceptible product' was included to promote creativity as an attribute that can be evaluated. The product referred not only to artifacts but to a variety of outcomes such as behavior.

In educational settings, creativity is often linked with open-ended learning situations in which the student needs to explore space [7]. In the definition of Plucker et al [6], this can be seen in the inclusion of both process and environment; for instance, consider brainstorming in a particular setting.

Epstein's [8] generative theory in the field of behavioral sciences yielded a definition of four core competences for creativity: broadening, challenging, surrounding, and capturing. These indicate that broadening an individual's knowledge base contributes to the individual's creativity. Moreover, creativity is stimulated by seeking challenge and changing environments, and it occurs frequently but needs to be captured. Due to the focus on behavior, emphasis is here not placed on a resulting product.

We should also mention 'flow', which is discussed as part of creative activity in the work of Csikszentmihalyi [9]. He outlined flow as a condition that enables an individual to tolerate obstacles and continue at the task.

Literature shows an agreement that creativity is found in and can be practiced by all of us; see, e.g., [6], [8], [10]. Herbert [10, p. 133], who studied creativity from a psychoanalytic perspective, concluded that "Creativity is inherent to all human beings and is not dependent on IQ or genetic inheritance." She continued that high IQ and genetic inheritance are likely to contribute to the ability to be creative but are not a necessary condition for it. Epstein [8, p. 766] summarized that "[...] with appropriate training, almost anyone will display a high degree of creativity."

B. Examples of Creativity in CSEd literature

CS teachers have employed various application contexts to enliven students' learning experiences and refer to creativity as one design attribute of such contextualizations. Examples include media computing, game programming, robots, and open data.

Media computing was designed to invite non-majors and women to study computing [11], [12]. This CS1 course design promoted creativity among other attributes that included collaboration, relevance, and hospitality of the study environment. Greenberg et al. [2] asked CS1 students to create visual designs using the Processing programming language, and observed that these students were more interested in investing their time on assignments and taking CS courses than students taking a traditional CS1. Using media computing as their setting, Porter et al. [4] asked students to manipulate pictures by programming and arranged an art show where student-prepared artistic images were presented. The goal was to emphasize the creative side of programming to the students and promote the study program in the campus. Actions of these kinds arguably make explicit associations between programming and arts.

Video games have been proposed to be used as CS1 course assignments because they allow students to develop individual solutions and therefore introduce creativity into the course [13]. Game programming is used as a contextualization for CS1 [14] as well as for outreach events [15].

The work by Apiola et al. [16] illustrates the use of robots in creative, open-ended assignments for intermediate students. These authors observed experiences of flow and rewarding problem-solving on the part of students who welcomed the open-ended assignment. Students who found the setting challenging complained about the course arrangements.

Open resources (Data, APIs) have been used as a resource for student-ideated projects [17]. In this setting also, third-year students' responses to the creative, open-ended project assignment were divided. Some students reported on internal motivation arising from self-ideated work, while others noticed that they were used to guided work and defined themselves as non-creative. The students also noted that the pressure to pass courses narrowed their investment in project work regardless of the opportunity to develop self-ideated products.

Furthermore, the locution 'computational creativity' is used in the literature when creative problem solving is referred to in the nexus of creativity thinking and computational thinking [18]. Exercises that are said to integrate creativity and computational thinking have been accordingly termed Computational Creativity Exercises (CCEs). These exercises seek to enhance learning in a way that students learn to apply and find solutions in new situations [19]. Soh et al. [20] summarized evidence on improved learning outcomes when CCEs were used. This exercise conceptualization leans on Epstein's core competences (see above) and, rather than discussing outreach and retention, seems to characterize creativity *needed* in CS problem solving and attempts to improve learning.

A potentially interesting observation is that creativity referred to in conjunction with an engaging application context for novices is constantly preferred by teachers and in the data examples provided from students. Subsequently in the curriculum, with more comprehensive assignments that require students to more substantively ideate what CS artifact is produced, students' reactions are nevertheless reported to be divided [16], [17]. Another interesting perspective is found

in Peters' [1] work on identity. She found that how students relate to CS become narrowed along with taking computing courses. Relatedly, in a master thesis work supervised by the present authors [21], it was found that creativity persisted as a key category in students' considerations through a CS1 period but other categories grew in quantity; we would speculate that considerations of creativity may become 'masked' as students focus on learning to program. These viewpoints suggest that how students relate to creativity should be studied in a longitudinal manner; the present study focuses on a starting point of such a continuum.

C. What is programming?

Several studies have looked at how students perceive programming: [22]–[27]. All these studies reported categories according to the phenomenographic or phenomenological approach. There seems to be at least an overlapping 'zone' that starts from a mechanistic, syntax, and programming language -related views of programming, and proceeds to the view of programming as problem solving. Some differences are found in what was identified as the least sophisticated category: Bruce et al. [22] reported a "following" category, which explains how students first make progress, whereas the results by Eckerdal et al. [23] began from a syntax- and language-related view. Other differences related to the most sophisticated categories reported. Moreover, the phenomenological approach raised aspects in the perceived condition of learning to program [26]. We speculate that the differences between the study results arise from the contexts and the perspectives included in the data collection measures. The phenomenological approach naturally indicates a lens different to the phenomenographic one. The present study began from a full thematic analysis of 40 data incidents, which yielded a set of categories that appears richer as compared to those reported in the studies above. We believe this was due to the timing of the data collection. This was before the academic courses of the first semester had really started and the data represents the students' 'visions' of programming in both narrow and large senses, with yet an undeveloped connection with the CS1 course they were about to study. The data contained several perspectives solely related to creativity.

III. Method

A. Data Collection

The data set consists of 1,946 student writings over the eight-year period 2010–2017. A single data incident is the collection of the student's textual answers to following questions about programming:

- What is programming?
- What is important in programming?
- What does it take to do programming?
- Describe what you like in programming.
- Describe what you do not like in programming.

The questions were originally designed to complement each other without a worry of overlap, to prompt a respondent to characterize programming in a rich way; each student's answers to these questions were combined into a single textual presentation before the analysis. During these years, the questionnaire was issued to CS1 students at the very beginning of each course instance. The course is offered bi-annually, meaning that the eight-year period covers 16 course instances. Students were expected to answer the questionnaire as soon as possible, even before the first lecture, to prevent any "pollution" of the conceptions. Response rates have not been systematically recorded, but they are quite high, most likely around 60–80% yearly.

The students reached by the CS1 pre-questionnaire are computer science and information systems students and all minors; there is no separate course for minors. The minors can be from any faculty of the multi-disciplinary university, while typically the majority of the minors come from science and statistics.

B. Procedure

The work was started by the first author who applied an initial content analysis step to a sample of 40 incidents, extracting all low-level perspectives that illustrated students' positions toward programming. This set of data was selected based on practical convenience; the incidents had been recently screened in another project, and immersion in these incidents had occurred already. More importantly, this set was checked to represent rich (wordy enough) incidents within the large data set, to be able to use this initial analysis as part of the ensuing analysis on the longest (hence, most elaborate) 240 incidents in the data. By reviewing the multitude of low-level codes, a decision was made to focus this study on a single perspective observable in several ways in the codes: the topic of creativity. Such a limitation appeared necessary resource-wise, while exploring this topic in particular was deemed potentially interesting for computing educators.

The whole data was sorted according to the length, and shared with the three authors: N=40 (the initial sample), N=100, N=100. The first author revised the initial analysis into a coding scheme—a set of categories—that was handed to the other two. In practice, the categories became the column titles in the sheet tabs of a spreadsheet file in which the data and the analyses were managed; a single sheet tab represented a shared "codebook." Thus, the authors first analyzed mutually different sets of data. These sets (N=40, N=100, N=100) were also independently coded by one other analyst. Afterwards, three sessions of 2–3 hours were arranged in which the three analysts negotiated agreement on each data incident where the two independent codings had differed. The data analyses were managed be devoting one sheet tab to each analyst, and one more tab was devoted to the peer-checked and negotiated final analysis. Throughout this process described, a single data incident could indicate more than one category in the coding scheme.

The initial coding step was inductive and relied on the theoretical sensitivity of the analyst. For example, taking into account 'flow' was based on the awareness of the connection between creativity and flow in the literature. Moreover, this

sensitivity included assumptions that student-expressed desires to create were rather direct indications of personal favoring of creativity in the given context of programming, and that comparisons to arts characterized programming as creative, artistic activity. In the agreement negotiations, any hesitations were managed by considering the literature presented in the previous sections, in particular the rich while concise definition of creativity by Plucker et al. [6]. What was discussed most was which perspective(s) of creativity were indicated. For instance, a data segment could be interpreted to demonstrate personal preference for freedom to create but also indirectly some other category such as creative innovation. Altogether, challenges were responded to by the exhaustive three-person reviews of the codings.

The number 240 was decided by considering resources: the goal was to analyze a considerable amount of qualitative data and yet to peer check the analyses. For the remaining, still a large segment of the data (1,976−240 = 1,706 incidents), we include an approximation of the portion of incidents that included one or more of the words or locutions that students typically used when referring to creativity within the sample of 240 incidents. This procedure is explained in Section IV-I.

IV. Results

Overall, we found that 59% of the respondents (in the sample of 240 respondents) conveyed creativity in their expressions in some way, while 41% did not. Of those expressions that conveyed creativity, we extracted eight qualitatively different categories that represent the aspects of how creativity showed in the texts. The categories are sorted in descending order based on their frequency of occurrence. Note that one respondent's text could represent multiple aspects; thus, a response that manifests creativity could fall into one or more categories. In the following, we present each of the emerged categories along with illustrative quotations. The categories are presented by first explaining a category as it was interpreted from the data by the researchers and then displaying illustrative quotation(s). The brackets within the quotations were added by the researchers to clarify the meaning of the translated (from Finnish to English) colloquial expressions to the reader.

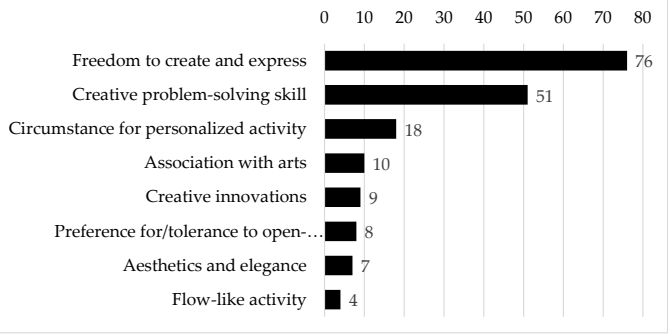

Fig. 1. Categories and their frequencies.

A. Freedom to create and express (n=76)

In this category, programming is experienced or envisaged as an activity that gives the actor freedom. This was the most prominent category; 54% of the answers that demonstrated creativity in some form were also included in this category. There were three central notions in how creativity was positioned: (i) freedom to create whatever the programmer likes, (ii) freedom to resolve unforeseen problems, and (iii) freedom to express oneself.

Freedom to create: Programming is enjoyable or fun because it allows creating whatever one wants. This freedom is not bound or restricted by anyone.

> Programming is one way to demonstrate creativity. Thinking that one can create something new and potentially useful is fascinating. (Student #49)

> In programming, I am more fascinated about the thing that you can create so much and whatever you like. The freedom to create in programming is the thing that made me study computer science. (Student #216)

> Creating something completely new and testing how it works is rewarding. (Student #202)

> Programming means to me certain kind of freedom. As you manage it, human can create what ever digital information with a laptop. (Student #165)

Freedom to resolve unforeseen or incredible problems: It is only our imagination that bounds the possibilities in what can be innovated, achieved, and created by programming. This notion was often expressed by emphasizing unlimited possibilities in programming.

> In my opinion, programming is the way to make incredible things real. (Student #235)

> [I like] the thing that imagination is the only limit as for what you can create. (Student #88)

One interesting observation in this category was that it illustrates how beginner programmers envisage, and almost fantasize about the do-goodism potential of the programming activity. At the beginning stages of their studies the learning curve is quite understandably left out of consideration.

> Programming gives one the freedom to resolve small and large problems with computer, even if [in cases] no one else has encountered the problem at hand. (Student #212)

Freedom to express oneself: Programming is a "language" for personal expression. On the other hand, programming is a way to observe and analyze others' expressions. Some respondents explicitly highlighted the possibility to express one-self while others depicted this more indirectly.

> The thing I like about programming is the granted freedom to express one-self. (Student #223)

> [I like] the freedom to apply, and the possibility to create a system of your own kind. (Student #186)

Personal expression can also be understood as making a personally relevant tool.

> Personally, progamming is an opportunity to develop [my] own tools, which have features that others have not considered necessary. [It's] one way of examining the world from my perspective. (Student #240)

B. Creative problem solving (skill) (n=51)

In this category, there were two main notions: (i) Programming is a skill, which is essential for solving complex problems creatively, and (ii) the task of programming is characterized by creative problem-solving. In the present data set, the matter of creative problem-solving was expressed both without and with explicit use of the word creativity. The former is illustrated by the following.

> If programming nevertheless is not working out, one needs the skill to question one's own logic for finding alternative conventions. (Student #218)

The illustration below includes explicit references to creativity.

> As a work, programming can be described as creative problem-solving. (Student #225)
>
> [Programming requires] creativity when encountering problems or technical constraints. (Student #214)

Respondents' perception of what solving problems creatively really means varied. Some saw it as applying "MacGyverisms" if needed, that is, using readily available components for reaching solutions within a short time limit. This is illustrated in the following quote.

> Creativity includes the skill of hacking, that is, implementing a thing as quickly and optimally as possible. (Student #86)

Others took a more analytical view and suggested that creativity stems from systematical considerations, and even needs stubborn patience to weigh out different perspectives to reach optimal solution.

> [In programming,] pedantry, logical thinking and creativity [are needed]; sometimes also stubborn patience is beneficial. (Student #99)

As an aside, mere immersion in the data indicated "problem solving" (the term was explicitly used many times in the data) as one of the major features associated within programming. The above explains specifically what is indicated with the term "creative problem solving."

C. Circumstance for personalized activity (n=18)

In this category, programming is characterized by positive feelings when the programmer identifies one's own hand in the artifact. Programming is seen as personal, and the urge for "I made it myself!" sensations are quite pronounced. The idea is that the construction of an artifact necessitates a personal, creative process.

> In programming, you can unleash your creativity, and you can see the result of your work very concretely. (Student #94)
>
> I don't have experience in programming, but I'm sure the most interesting will be the actual implementation, that is, experiencing and seeing my hand concretely. (Student #137)

D. Associations with arts (n=10)

This category shows in two ways: (i) the first is the general characterizations of programming as artistic work, and (ii) the second is students explaining how programming associates with specific artistic activity or abilities. The former is illustrated by the quotations below, with the third quotation further linking programming to human consciousness.

> [...] Designing the program and understandings of [its] larger contexts require societal and artistic vision (Student #23)
>
> I like that programming is creative and artistic work, that you identify the writer, although the code itself would follow an expected format. (Student #220)
>
> Nowadays It's became both art and craft, being a transhumanist, I am convinced that programming in the future will become an element of consciousness. (Student #235)

Artisic activies included drawing, poetry, and tinkering with Legos. In the example below, the student compares learning to program with knitting and learning to play the piano.

> It is bit like knitting a scarf but with a computer [...] I attempt to relate to this course similarly to piano playing, which I have rehearsed ten years: rehearsing is comprehensive and time consuming, but not necessarily rocket science. (Student #118)

This quotation concurrently illustrates how the learning challenge integral to programming is cognitively tolerated by an analog to a familiar activity. The artistic abilities included a sense of drama, musicality, and visuality.

E. Creative innovations (n=9)

Students seemed to be motivated by the possibility of innovations. Such indications occurred both explicitly and implicitly. The former is illustrated below.

> I understand that, in the end, it is a creative activity [...] I am highly interested in the possibilities for new innovations that programming makes possible. (Student #215)

The implicit occurrences rather typically showed interest in doing something socially useful:

> I like the thing that by programming one is able to do significant things for the benefit of people's every-day life. (Student #48)

F. Preference for/tolerance to open-ended situations (n=8)

This category comprises depictions where programming is (i) acknowledged as an open-ended challenge where no correct answers are provided to the programmer, and no single, "one-size-fits-all" solutions exist. We paid particular attention to (ii) the students' tolerance to this condition.

Examples showing acknowledgment (i) are given below, the latter highlighting agility needed to manage ever-evolving environment.

> One is challenged, because one has to figure out each solution, and nobody gives ready-made answers. (Student #204)

> [Programming requires] ability to transform [...] One really can't rote learn (at least not many things). There's neither just one right way of making a particular program. (Student #135)

It was interesting to notice that students' responses to open-ended situations, which they assigned to programming, diverged (ii). The other was acknowledging this feature while emphasizing the easiness of (and implicitly preference for) just selecting one approach:

> Things can be done in many ways, but the easiest way out is to select one and adhere to it. (Student #213)

In place of creativity, this emphasizes finding a way out of an open-ended situation. Other considerations revealed a respondent's personal tolerance to, and preference for, this openness. In the example below, this interpretation was further reinforced by the same respondent showing personal preference for programming allowing for creativity.

> I consider it interesting how, in talking about programming, you do not necessarily have a 'one and only' solution for solving and implementing a particular thing. You can also utilize your own creativity I suppose, which to my eye is a positive thing. (Student #219)

G. Aesthetics and elegance (n=7)

Both the students (i) with and (ii) without obvious pre-experience in programming referred to aesthetics and elegance, which is illustrated below, respectively.

> I appreciate optimized and hacked code, so I obviously consider these aspects important. [I like] admiring and presenting finished code. (Student #87)

> In the future, when I understand programming, I very likely begin to identify and hate bad code (Student #125)

The previous quotations were implicit locutions indicating the category of elegance and aesthetics. The quotation below shows enjoyment out of elegance, with an explicit reference to it.

> The bonus is the warm feeling in your heart engendered by elegant code [...] I like finding stylish solutions. (Student #213)

H. Flow-like activity (n=4)

Some parts of the literature associate flow with creative activity (see Section II-A), an aspect also found in the students' considerations. Such considerations originated from the students who seemed to have experience in programming before starting CS1, as is obvious in the example below.

> [...] Thereby, for instance insufficient interface documentation readily destroys the programming flow, which in itself is an integral element in the process of programming. (Student #212)

The example below further explains that flow may be seen as a desirable state in programming activity.

> [..] Also, being constantly stuck, that you are not even close to a flow-like sate in which your personal skills/understanding and the challenge of programming intersect, can be frustrating. (Student #238)

I. Approximation for the remaining of the data

For the remaining data (1,976−240 = 1,706 incidents), we state here an approximation of the portion of incidents that included one or more of the words or locutions that students typically used when referring to creativity within the sample of 240 incidents. This intends to cover the whole data set for an overview, with an acknowledgment that this part of the analysis is approximative. The analysis was a combination of using the `grep` command line utility and human inspection to count in only relevant hits. The search words identified relevant were different forms and inflections of Finnish words comparable to the English words freedom, creativity, opportunity, imagination, what ever, one-self, new, one's own, art, elegant, aesthetics, beauty, flow, innovation, figure out, express, discover — at best a simplified explanation due to the differences between the languages.

The analysis was done by the analyst inspecting the results of each `grep` application, and removing hits from this data segment step by step, before the next search word was used. When all the search words were used, 72 % of the data incidents were left, meaning that a total of 28 % of the data incidents (students) provided an indication of creativity that reflected the categories reported in the previous sections. It is important to note that all uncertain indications that would have required a more detailed analysis than was possible with the one-line context returned by `grep`, or were considered to benefit from peer checking, were not counted as hits. For instance, if the one-line response included references to 'opportunities' without enough context to decide whether this actually denoted employment opportunities or creative opportunities, the data item was not considered a hit. Therefore, we can approximate that *at least* 28 % of this data segment included references to creativity, but also that a more detailed human analysis would have likely revealed a larger portion. Given that the students' answers in this segment of the data got shorter (less elaborate), we speculate that this amount reflects a similar tendency that we observed in the carefully peer-checked sample of 240 items.

V. Discussion

This exploratory research looked at how and to what extent creativity was indicated in the considerations of students who were about to study CS1. If we return to look at the nature of the categories (Figure 1), we can identify two principal ways in which the students referred to programming in reference

to creativity: personal and characterizing. This division is illustratively present in the two largest categories: "Freedom to create and express" reflects students' *personal* desires, whereas "Creative problem-solving" indicates students *characterizing* programming as problem-solving in which creativity is needed and to which it inherently belongs. Merely by looking at the frequencies, this division seems to differentiate the respondents rather well: a total of 61 out of the 76 students who indicated the freedom category, did *not* simultaneously indicate the problem-solving category. It might be that students who personally prefer creativity focused their consideration on this personal position.

The category "Circumstance for personal activity" is also clearly about a *personally* valued experience. By selecting those who indicated the freedom category or this circumstance category (or both), a total of 90 out of 240 (our main research sample) can be interpreted to convey a personal, favoring, position on creativity. This exploratory research, applied to an existing data set in which creativity was not explicitly probed, does not have a reference to judge what is a "sufficient" number of references to claim that creativity is prevailing. In our speculations, this high number of personal positions alone documents support for the actions many an educator has taken; consider the creative settings in introductory programming that promote engagement and enable personalization of learning.

The literature suggests that part of students may be troubled when creativity is presumed in authentic settings, such as open-ended projects, later in the curriculum [16], [17]. In such settings, the students are required to develop ideas for what artifact is produced altogether [17] or propose technological visions for customers starting from a highly ill-formed settings [28]. Perhaps relatedly, Peters [1] foregrounded a potential narrowing influence of taking computing courses on students' identities, whereas Isomöttönen et al. [17] reported that the education system had likely affected some students' preparedness for open-ended settings. When we look at the present data beyond what was reported, we see that it highlights irritating experiences with meticulous and tedious programming work. In effect, this obvious point was present in the illustrations of flow (Section IV-H). On these grounds, we would formulate that one challenge is the distance between a creativity vision and the feasibility of that vision in programming. We propose longitudinal studies that are grounded in the creativity expectations documented here and identify the multitude of aspects that come on the way of students' liking of freedom to express and create. This research should also identify students with unarticulated personal positions on creativity, and track the roots and persistence of those positions.

Several other future research targets can be identified and were also raised by insightful reviewers. Beginner students' creativity expectations could be studied across academic disciplines. Thus, to develop a big picture of creativity expectations, it would be interesting to see if such expectations are provoked by or tightly integrated with a topic such as programming. Further content analysis could be conducted to explore if any changes can be observed in beginner students' expectations over a several-year period. Another kind of goal would be to further clarify how beginner programmers see their potential route to becoming programmers as artists. Yet another interesting topic would be to look at how students' epistemological beliefs or fixed-vs-growth mindsets relate to their creativity expectations; in the present analysis, for instance the category of Tolerance to open-ended situations can be regarded as an indication of a stance toward knowledge that is not fixed.

REFERENCES

[1] A.-K. Peters, "Learning computing at university: Participation and identity: A longitudinal study," Ph.D. dissertation, Acta Universitatis Upsaliensis, 2017.

[2] I. Greenberg, D. Kumar, and D. Xu, "Creative coding and visual portfolios for CS1," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '12. New York, NY: ACM, 2012, pp. 247–252. [Online]. Available: http://doi.acm.org/10.1145/2157136.2157214

[3] M. Guzdial, "Exploring hypotheses about media computation," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ACM. San Diego, CA: ACM, 2013, pp. 19–26. [Online]. Available: https://doi.org/10.1145/2493394.2493397

[4] L. Porter and B. Simon, "Fostering creativity in CS1 by hosting a computer science art show," *ACM Inroads*, vol. 4, no. 1, pp. 29–31, 2013. [Online]. Available: http://doi.acm.org/10.1145/2432596.2432609

[5] R. J. Sternberg, *How to Develop Student Creativity*. Alexandria, Va: Association for Supervision and Curriculum Development, 1996.

[6] J. A. Plucker, R. A. Beghetto, and G. T. Dow, "Why isn't creativity more important to educational psychologists? potentials, pitfalls, and future directions in creativity research," *Educational Psychologist*, vol. 39, no. 2, pp. 83–96, 2004. [Online]. Available: https://doi.org/10.1207/s15326985ep3902_1

[7] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, T. Selker, and M. Eisenberg, "Design principles for tools to support creative thinking," Tech. Rep., 2005.

[8] R. Epstein, "Generative theory," in *Encyclopedia of Creativity*, M. A. Runco and S. R. Pritzker, Eds. San Diego, CA: Academic Press, 1999, vol. 1, pp. 759–766.

[9] M. Csikszentmihalyi, *Creativity: Flow and the Psychology of Discovery and Invention*, 1st ed. New York, NY: HarperCollins Publishers, 1996.

[10] A. Herbert, *The Pedagogy of Creativity*. New York, NY: Routledge, 2010.

[11] M. Guzdial, "A media computation course for non-majors," in *ACM SIGCSE Bulletin*, vol. 35, no. 3. ACM, 2003, pp. 104–108. [Online]. Available: https://doi.org/10.1145/961511.961542

[12] L. Rich, H. Perry, and M. Guzdial, "A CS1 course designed to address interests of women," in *ACM SIGCSE Bulletin*, vol. 36, no. 1. ACM, 2004, pp. 190–194. [Online]. Available: https://doi.org/10.1145/1028174.971370

[13] M. T. Morazán, "Functional video games in the CS1 classroom," in *International Symposium on Trends in Functional Programming*. Springer, 2010, pp. 166–183. [Online]. Available: https://doi.org/10.1007/978-3-642-22941-1_11

[14] V. Isomöttönen, A.-J. Lakanen, and V. Lappalainen, "K-12 game programming course concept using textual programming," in *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY: ACM, 2011, pp. 459–464. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953296

[15] A.-J. Lakanen, "On the impact of computer science outreach events on k-12 students," *Jyväskylä studies in computing*, no. 236, 2016. [Online]. Available: http://urn.fi/URN:ISBN:978-951-39-6634-8

[16] M. Apiola, M. Lattu, and T. A. Pasanen, "Creativity-supporting learning environment—CSLE," *Trans. Comput. Educ.*, vol. 12, no. 3, pp. 11:1–11:25, Jul. 2012. [Online]. Available: http://doi.acm.org/10.1145/2275597.2275600

[17] V. Isomöttönen, M. Daniels, A. Cajander, A. Pears, and R. McDermott, "Searching for global employability: Can students capitalize on enabling learning environments?" *ACM Transactions on Computing Education – Special Issue on Global Software Engineering Education*,

vol. 19, no. 2, pp. 11:1–11:29, Jan. 2019. [Online]. Available: http://doi.acm.org/10.1145/3277568

[18] L. D. Miller, L.-K. Soh, V. Chiriacescu, E. Ingraham, D. F. Shell, S. Ramsay, and M. P. Hazley, "Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses." in *IEEE Frontiers in Education Conference (FIE)*. IEEE, 2013, pp. 1426–1432. [Online]. Available: https://doi.org/10.1109/FIE.2013.6685067

[19] M. S. Peteranetz, A. E. Flanigan, D. F. Shell, and L.-K. Soh, "Helping engineering students learn in introductory computer science (CS1) using computational creativity exercises (CCEs)." *IEEE Transactions on Education*, vol. 61, no. 3, pp. 195–203, 2018. [Online]. Available: https://doi.org/10.1109/TE.2018.2804350

[20] L.-K. Soh, D. F. Shell, E. Ingraham, S. Ramsay, and B. Moore, "Learning through computational creativity." *Commun. ACM*, vol. 58, no. 8, pp. 33–35, 2015. [Online]. Available: https://doi.org/10.1145/2699391

[21] H.-J. Niemi, "Opiskelijoiden käsitykset ohjelmointiin," Master's thesis, 2014, in Finnish.

[22] C. Bruce, L. Buckingham, J. Hynd, C. McMahon, M. Roggenkamp, and I. Stoodley, "Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university," *Journal of Information Technology Education: Research*, vol. 3, no. 1, pp. 145–160, 2004. [Online]. Available: https://www.learntechlib.org/p/111446/

[23] A. Eckerdal, M. Thuné, and A. Berglund, "What does it take to learn 'programming thinking'?" in *Proceedings of the First International Workshop on Computing Education Research*, ser. ICER '05. New York, NY: ACM, 2005, pp. 135–142. [Online]. Available: http://doi.acm.org/10.1145/1089786.1089799

[24] I. Stamouli and M. Huggard, "Object oriented programming and program correctness: The students' perspective," in *Proceedings of the Second International Workshop on Computing Education Research*, ser. ICER '06. New York, NY: ACM, 2006, pp. 109–118. [Online]. Available: http://doi.acm.org/10.1145/1151588.1151605

[25] M. Thuné and A. Eckerdal, "Variation theory applied to students' conceptions of computer programming," *European Journal of Engineering Education*, vol. 34, no. 4, pp. 339–347, 2009. [Online]. Available: https://doi.org/10.1080/03043790902989374

[26] C. Rogerson and E. Scott, "The fear factor: How it affects students learning to program in a tertiary environment," *Journal of Information Technology Education: Research*, vol. 9, pp. 147–171, January 2010. [Online]. Available: https://www.learntechlib.org/p/111361

[27] A.-J. Lakanen and V. Isomöttönen, "What does it take to do computer programming?: Surveying the k-12 students' conceptions," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 2015, pp. 458–463. [Online]. Available: https://doi.org/10.1145/2676723.2677229

[28] M. Daniels, Å. Cajander, A. Pears, and T. Clear, "Engineering education research in practice: Evolving use of open ended group projects as a pedagogical strategy for developing skills in global collaboration," *International journal of engineering education*, vol. 26, no. 4, pp. 795–806, 2010.