Value-Focused Investigation into Programming Languages Affinity

Alvaro Costa Neto
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Barretos, Brazil

Cristiana Araújo
Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar – Braga, Portugal

Maria João Varanda Pereira
Research Centre in Digitalization and Intelligent Robotics,
Polythechnic Insitute of Bragança, Portugal

Pedro Rangel Henriques
Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar – Braga, Portugal

Abstract

The search for better techniques to teach computer programming is paramount in order to improve the students' learning experiences. Several approaches have been proposed throughout the years, usually through technical solutions such as evaluation systems, digital classrooms, interactive lessons and so on. Personal factors, such as affinity, have been largely unexplored due to their qualitative and abstract nature. The results of a preliminary survey on how and why affinity is created between programmers and their favorite languages, conducted on a master's degree class at Universidade do Minho, showed unexpected results as to which languages became favorites and the possible reasons for the students' choices. Aiming at further exploration on this topic and continuation of this research, the Value-Focused Thinking method was applied in order to construct a more complex, in-depth survey. This value-oriented method kept focus under control and even raised a handful of opportunities to improve the research as a whole. This paper describes the Value-Focused Thinking method and how it was applied to construct a new and deeper computer programming education survey to understand affinity with languages.

2012 ACM Subject Classification Social and professional topics → Computing education; Software and its engineering → General programming languages

Keywords and phrases Computer Programming, Programming Languages, Affinity, Education, Learning, Value-Focused Thinking

Digital Object Identifier 10.4230/OASIcs.ICPEC.2022.1

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/05757/2020 and UIDB/00319/2020.

1 Introduction

Being an inherently intricate process, learning computer programming is widely accepted as being a complex and difficult task. Technical approaches applied to research on how to teach and learn computer programming date many decades ago [6, 1, 2], and has kept high interest in the academic field [7, 18, 14, 3, 9, 10] ever since. Personal and contextual factors also play important roles in learning and should be considered when teaching computer programming. Pedagogical research has shown for more than a century that these personal factors are influential to the teaching-learning process [12, 17, 8, 4] and should be taken into account at all times.

Affinity to programming languages, as a personal factor, may play an important role in the teaching-learning process, as indicated in a previous survey conducted in 2021 with a class of master's degree students [13]. Those preliminary results showed that affinity is more complex than previously assumed and that a deeper, more structured study should be conducted. A value-focused approach was used in order to better organize the construction of the survey for this new study, yelding a strongly focused questionnaire.

This paper starts with a brief presentation of a former preliminary study on affinity to programming languages, but focuses on the construction of a new survey through the Value-Focused Thinking method. The new survey will be used in the near future, aiming to further understand how affinity to a programming language is created. The overall structure of this article is composed of six sections: the introduction discusses which factors may influence the learning process and how they may be categorized. The second section presents the initial investigation, including a preliminary study about affinity to programming languages and a small previous survey that showed interesting results. The third section gives a general explanation of Value-Focused Thinking (VFT), a decision making method used to structure the new survey. Section four explains how Value-Focused Thinking was directly applied to the construction of the new survey. Section five lists the final structure of the new survey and the expected results. The final section concludes this paper and lists the next steps that shall be taken to apply the new survey on affinity to programming languages.

2 Background and Previous Work

Assuming that personal factors are relevant to students – as previously stated – and focusing on the affinity that is commonly observed among programmers (both students and professionals) towards specific programming languages, it would be reasonable to investigate how it is established and which role it plays on the learning process. In order to initiate this investigation, a preliminary study was conducted.

2.1 Lecture and Preliminary Study

In order to better understand what role affinity takes in the learning process and how it is constructed, a preliminary study was conducted with twenty three students of a Masters' degree class in Computer Engineering at Universidade do Minho [13]. The study consisted of a lecture about teaching and learning computer programming, a quick survey during the lecture, and a small questionnaire with a few questions about programming languages, their learning experiences, and which languages they prefered.

The lecture presented and discussed which factors might be relevant to the learning process and how these factors could be applied to improve the students' experiences. During the lecture, a quick survey was conducted, based on snippets of source code that printed a few numbers of the Fibonacci sequence. These snippets were written in seven different languages – BASIC, Lisp, C, Java, Python, Ruby, and Swift – and were shown sequentially at first, and then simultaneously for comparison purposes.

Since it would be impractical – and probably confusing – to show all possible combinations, only a few comparisons were made. Each combination tried to explicate either differences or similarities in their programming languages, encouraging students to consider unusual characteristics that could influence their affinity to one of the languages. As an example, when comparing two different versions of Lisp code, the goal was to highlight the impact caused by the use of the natural coding style for a certain language. This would allow the students to consider nuances beyond the plain syntax definition of a language.

Students were then promptly asked to vocalize their preferred languages and the answers showed C, Java and Haskell[1] as the three highest ranked choices. This question was proposed and read during the lecture, and students answered directly without any kind of written form.

After the lecture, the students were then asked to answer a small questionnaire containing three questions:

1. In a range of *very low* to *very high*, how important is the language choice for learning computer programming? Justify your answer;
2. Which factors – presented in the lecture – are most relevant and influential to learning computer programming?
3. Which languages would you choose to teach computer programming: BASIC, Lisp, C, Java, Python, Ruby, Swift, or some other language? Justify your answer.

The answers to the first question were interesting, albeit inconclusive. While almost every student agreed that it is crucial to wisely choose an initial programming language, pretty much all of them differed on the justification. Opinions ranged from technical – mainly based on the availability of certain syntax constructs and data structures – to pedagogical – the initial language should have an easy learning curve in order to avoid discouraging students. Other justifications cited documentation availability, prospective employability, and in one case, indifference for the language choice *per se*.

Answers to the second question focused on the obstacles that the initial programming language could impose. Technical aspects of the language were preeminent among the answers, but *affinity* was also considered an important factor. Other personal factors included: the relationship stablished with the teachers and the motivation that is cultivated during the first contact with computer programming.

The last question allowed multiple choices and its main goal was to compare the results of the lecture's question (the one where C, Java and Haskell were three favorite languages), with the selection of languages they would choose to pass on, possibly influencing the affinity other students would develop. This would either support their initial choices – their preferred languages were chosen to continue the learning cycle – or contradict it – their preferred languages and their choices to pass on programming knowledge were different. The final goal for this question was to test for external influences, such as popularity and market share. Being currently a popular programming language [16, 15], with many applications in high demand, such as artificial intelligence, data science and numerical computing, Python was expected to be preferred. Nonetheless, C and Java were the highest ranked in the questionnaire, while Haskell also had a perceptive presence in the results and tied the fourth place with BASIC (figure 1). These results become clearer if the students' affinity to these languages was actually constructed as a *consequence of their learning experiences*, since they had been formally taught Haskell, C and Java as their initial programming languages.

These results implied that more conclusions would arise from a new survey for a deeper investigation on how students learn computer programming and their prefered languages. The complex nature of personal factors that influence this process creates opportunities for continued investigation on *affinity with programming languages and its relation to the learning process.*

[1] Haskell is the first programming language for students of Universidade do Minho.

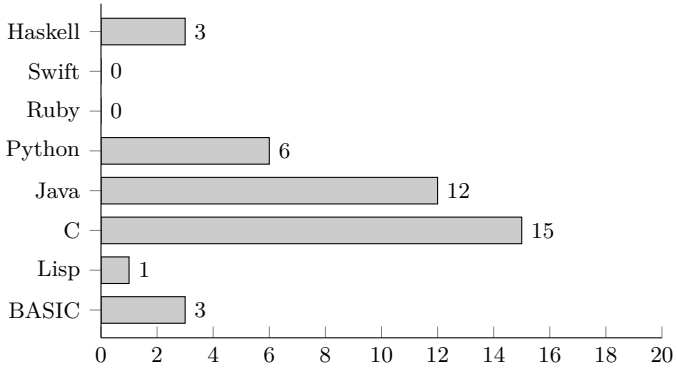

Figure 1 Languages chosen by students when asked which one they would use to teach computer programming. These answers were gathered through a small survey after the lecture. This figure was originaly published in [13].

3 Value-Focused Thinking

Value-Focused Thinking (VFT) is a decision making process proposed by Ralph Kenney [11] and it was chosen as the formal method to plan and guide the construction of the new survey. It is based on the fact that planning alternatives and practical details in the first place tends to diverge the solution from what should be its main focus, concentrating efforts on features that might be discarded, and missing other opportunities that could emerge. In order to avoid this kind of recurrent and short sighted behavior, Kenney proposed a method that would force the definition of the main values and their derived objectives first, driving the whole decision making process with focus. In the author's words:

> Alternative-focused thinking is designed to solve decision problems. Value-focused thinking is designed to identify desirable decision opportunities and create alternatives [11, p. 538].

The method stablishes a well-founded process, based on the premise that the main values, central to the decisions being made, should always be enforced and guarded when thinking about objectives and alternatives. The whole VFT process follows three main stages: definition of values, gathering of objectives and construction of alternatives.

3.1 Definition of Values

The first stage in a Value-Focused Thinking process is to define the values and contextualize the problem to be solved. This step takes into account the desired results, what are the expectations – both for success and failure – and what kind of experience has been observed in the same context and on similar problems. While it is not the most operational part of the process, information gathered at this point is crucial to gather realistic objectives in the next step and to keep the whole decision making process focused.

3.2 Gathering of Objectives

In the second stage of the process, the objectives are gathered, sorted and classified. This stage is crucial for grounded and efficient construction of alternatives – the main goal of the Value-Focused Thinking process. The first to be defined is the *strategic objective* which states the main abstract goal of the decision being made. Other objectives are listed and roughly classified into *fundamental* or *means*:

- Fundamental objectives stablish the main reasons for the decision making in the first place, and are usually directly related to the strategic objective;
- Means objectives define what is necessary to achieve other objectives.

In order to determine if an objective should be considered fundamental or means, Kenney proposed the question "why is this objective important in the decision context?". If the answer is "because it is one of the essential reasons for interest in the situation" then the objective should be considered fundamental. On the other side, if the answer resembles "because of its implications for achieving some other objective", it is a means objective [11].

When the first version of the objectives list is concluded, revisions should be made in order to simplify – by aggregating redundant objectives – and reclassify the list – by applying the question above. There is no definition as to when this revision cycle should end. In summary, as soon as all important objectives are listed and classified, and the revisions no longer change the list, this part of the process is done.

3.3 Construction of Alternatives

The final stage concludes the process by creating alternatives while also identifying opportunities. The alternatives represent the courses of action to be taken and need to directly relate to the previously listed objectives (usually *means objectives*, but not exclusively). The most obvious alternatives usually come from previous experiences and commonly are the first ones to be thought of. Once these have been considered, deep thought about the problem should be carried on, in order to pursue hidden and more unexpected alternatives, always keeping in mind the values defined as per the Subsection 3.1.

Since each alternative should be related to at least one objective, the most straightforward way to undertake this stage is to list the objectives and propose one alternative for each. Once all objectives are evaluated, the construction stage restarts considering two objectives at a time, then three, and so on until all objectives are grouped together to create one alternative. This final state may not be reachable depending on the problem being tackled, but the construction stage should go as far as possible in this direction. Once all alternatives are listed, a review process should try to eliminate redundancies, which usually occurs with the first entries of the list.

Opportunities arise when trying to create alternatives. In some situations, an alternative presents some kind of limitation or necessity that must be fulfilled. Instead of considering it a failed attempt, one should identify these obstacles as opportunities to be further explored, possibly starting entirely new decision making processes.

4 Application of Value-Focused Thinking

The application of Value-Focused Thinking into the development of the survey was based on the fact that, for all intended purposes, *deciding which questions to ask and how to ask them is a decision making situation.* The choice of VFT among other methods was also motivated by its lean and straighforward mechanism that generates highly focused outcomes, and by previous positive experiences using it on similar projects.

The process of constructing the survey through VFT followed the standard course of action for the method: definition of the main values and context, gathering of the objectives, and construction of the alternatives[2].

[2] A read-only copy of the VFT document may be found in the following address: `https://bit.ly/3LrVV2D`.

4.1 Values and Context

The initial part of the VFT method is essential for focusing the rest of the process (as previously mentioned in Subsection 3.1) and it is usually a relatively straightforward part. In this case, though, the method was not being applied to a conventional situation – business related decision making, such as which parts to buy, who to buy them from or which bonds to sell – but to aid in the construction of a research survey. This peculiarity posed an interesting view on the whole process, since the decision being made did not apply directly to the research conducted through the survey, but *to the survey itself*. Being a means to an end – finding out which characteristics are related to programming language affinity – the survey is still part of the research as a whole, but the VFT was applied specifically to support the construction of the survey and its values represented that intent.

The chosen values were:

1. High comprehensability;
2. Focus on the main topics being researched;
3. Maximum coverage of different personal profiles;
4. Easy publishing and completion;
5. Gathering of valid and trustworthy answers.

The list of values clearly states the focus of the survey: to be *efficient* (values 1, 3 and 4) and *reliable* (values 2 and 5). Obvious as it might seem, it is crucial to list – and later abide to – these values. When thinking only on the objectives, as an example, one could easily lose focus and plan too many questions. By clearly stating the values, this unintentional mistake would be immediately identified violating "Easy [...] completion" and properly fixed.

The following step in this part was to list the perfect, average and terrible scenarios that could happen, to serve as guidelines. The perfect scenario pointed to a totally efficient questionnaire, with as many answers as possible, from multiple and varied sources, leading to a clear and encompassing conclusion. The average scenario presented high efficiency to the questionnaire with many answers from many sources, leading to an important and relevant conclusion. The terrible scenario represented an ineficient questionnaire with almost no answers, leading to no conclusion at all.

The final step for the contextualization listed previous experiences in similar endeavours that influenced the results of previous surveys and could possibly happen again. The main occurrences were:

- Low quantity of answers, which diminished the representativeness of the conclusions;
- Discarded answers that pointed to some kind of misunderstanding of what was asked;
- Unexpected and interesting results from open ended questions;
- Direct questions that seemed to be answered randomly (in contrast with other answers).

These experiences, being good or bad, were important warnings of caution to take into consideration for the next part of the VFT method: gathering and listing the objectives.

4.2 Objectives

Based on the values and the context previously stablished, the objectives were listed and categorized. The strategic objective, as explained in Subsection 3.2, represented the main goal of the survey: *to construct and conduct a capable, valid and trustworthy survey to evaluate which factors influence the affinity stablished between programmers – of any level or context – to computer programming languages.*

While the strategic objective was an abstract take on the main values, the following fundamental objectives were a further step into its concretization:

1. To select as wide a range of respondents as possible, including those without prior knowledge of computer programming;
2. To ask questions that allow finding correlations between respondents and their favorite languages;
3. To faithfully characterize both personal aspects of the respondents and technical aspects of the programming languages.

The next step was to stablish the means objectives. They were constructed to support the fundamental objectives, creating the basic ideas of a real survey. Albeit being the most practical step of the gathering of objectives, these should not include implementation details, as these would follow in the construction of alternatives. The mean objectives listed practical needs (an on-line survey system that allows the construction of the intended questionnaire), publishing strategies (educational institutions, social media and professional hubs), which kind of information to gather (personal data, educational history, which languages are known etc.) and interesting results to be obtained (influence of peers on the choices of favorite languages, favorite technical aspects of the programming languages, popularity etc.). These objectives were then directly mapped to an implementation proposal of the survey in the last part of the VFT method: the construction of alternatives.

4.3 Alternatives

The last part of the VFT method created the alternatives for the undergoing decision-making process. In a general sense, the alternatives presented different possibilities to achieve the intended goals. In this specific case, alternatives stablished different ways of constructing the questionnaire, always keeping objectives and values in mind[3].

Each element of the survey definition – which on-line survey tool to use, possible publishing methods, valid and robust user agreement, questions, and desired results – was directly related to the means objectives gathered in the previous part. This meant that the whole survey construction was indirectly guided by all of its values through its previsouly listed objectives.

As an example, one of the means objectives specified that it would be important to gather if the respondent has got a *competitive or cooperative nature*, in order to later verify a possible correlation between this kind of personal characteristic and his or her favorite languages. This objective resulted directly in the following question definition:

- **Subject:** competitive or cooperative profile;
- **Alternatives:**
  1. Indirect observation by asking the number of participations in competitive or cooperative activities;
  2. Direct question of personal preference – competitive or cooperative.

As can be seen, the question itself was not written in this stage, only its specifications were listed. This precaution was taken in order to avoid prematurely fixing the survey before it could be reviewed as a whole. As for the alternatives, they presented themselves as different ways of constructing the questions and which kind of answers would be allowed. The resulting structure of the survey is described in Section 5 with all question definitions that were proposed at this moment.

[3] It is important to realize that the term *alternatives* should be understood as defined in the VFT method, not necessarily options for answering a question. As an example, a compilation of available on-line survey tools could be considered alternatives.

During the definition of the questions, desired statistical results – listed in Subsection 5.4 – were also specified. These served as checkpoints for the questions themselves, revealing possible "blind spots" in the survey through the absence of questions that would be necessary to obtain certain results.

The final step in the creation of alternatives was to list the opportunities that have been identified so far. In total, three opportunities were listed:

1. Since the range of desired results and questions is very wide, it would be possible to create more than one survey, dividing and better focusing the research in specific areas, such as personal factors, technical characteristics of the languages etc.
2. Available free survey tools seem to be incapable of constructing more complex surveys, so it would be beneficial to create a new system;
3. Instead of using real languages to verify the influence of technical characteristics on affinity, it would be better to create a pseudo-language flexible enough to be used in all situations.

It is important to emphasize that *these opportunities were not mandatory courses of action*, but presented different decision-making possibilities that emerged during the process of the survey construction. They may not have been detected if the alternatives (the questions of the survey, in this case) were the first elements to be thought of and, ultimately, this is one of the main advantages of the Value-Focused Thinking method.

5 Structure of the Survey

After applying all the steps of the VFT method, the structure of the survey was completed albeit not implemented. The implementation – i.e. to add the sections, questions and commentary to an on-line tool for publication and participation – is a direct consequence of the planning process, much like in computer programming, and should not be considered a requirement for conclusion of the VFT method. Based on the constructed alternatives, a general structure for the survey has been reached and it was divided into three sections: *personal data*; *background and projections in computer programming*; *affinity to different programming language characteristics*.

5.1 Personal Data

This first section of the survey deals with personal background and is composed of nine direct questions:

1. Age;
2. Gender;
3. Country of residence;
4. Native language;
5. English language level according to the Common European Framework of Reference [5];
6. Formal education level, from "Uneducated" to "Doctorate or beyond";
7. Learning style, with choices for both easiest and hardest to learn: "Mathematical and numerical problems", "Logic exercises", "Memory based questions" and "Practical applications";
8. Household income *per capita*;
9. Current occupation.

The answers to these questions will be correlated to the programming language affinity choices in the third section of the survey (5.3).

5.2 Background and Projections in Computer Programming

This section deals with previous experiences and future goals specifically about computer programming. It is composed of seven questions:

1. Time spent studying computer programming, in years;
2. Time spent working at computer programming, in years;
3. Time spent teaching computer programming, in years;
4. Learning methodologies during studying computer programming;
5. Learning methodologies applied to teach computer programming;
6. Competitive or cooperative preference for group working, measured through number of participations in activities of each kind;
7. List of effectively known programming languages, multiple choices allowed;
8. Which programming language first had contact with;
9. Intended position for future jobs in computer programming.

The answers to these questions in this section will help on drawing a picture of experience with computer programming and its languages. The answers will be compared to choices made in the third section of the survey, in order to procure possible correlations of affinity to the respondents' backgroung with computer programming – meaning that affinity is a result of learning or working with a particular language – and personal foresight – meaning that factors such as popularity and market influence are relevant to affinity.

5.3 Affinity to Different Programming Language Characteristics

The last section of the survey deals directly with affinity, collecting data about which languages lead to affinity and why. It is composed of five questions:

1. Comparisons of source code snippets, written specifically to test common syntax and semantic differences in current programming languages. This question was divided into subquestions which are detailed below;
2. Affinity level to programming languages, measured from "No affinity at all" to "Favorite";
3. Change in affinity to programming languages, which ones lost affinity, which ones gained affinity thoughout the years and what was the perceived cause to the change;
4. Influence of peers in affinity, as a personal observation;
5. Main motivation for affinity to the favorite languages, also as a personal observation.

A list of programming languages has been selected to compose the questions 2 and 3 of this section. This list included both current and former popular languages, aiming also at gathering as varied characteristics as possible, such as syntax, semantics, market share, popularity, paradigm etc.

The snippets of source code shown in question 1 were written carefuly to contrast only one syntactical or semantical characteristic at a time. This question is essential for obtaining insight into which practical characteristics of the programming languages are influential to affinity growth. Also, it represented the most practical and applied questioning of the survey. The following characteristics are queried:

- Variable declaration syntax;
- String representation and basic operation;
- Type inference and conversion;
- Block delimitation;
- Conditional structures;
- Repetition structures;

- Function or method calling convention;
- Presence and use of jumps (as in *goto*);
- End-of-statement syntax;
- General paradigm;
- Default data structures;
- Verbosity.

This list covers most technical characteristics of programming languages that might have some effect on affinity. In order to avoid blurring the respondents' answers by other personal factors, the snippets were written in a neutral algorithmic language that was informally defined[4].

Answers in this section are paramount for any conclusions about affinity, since most of them will be used as the counterpart to the correlation with answers in the previous sections. In the end of the process, since this question would have too many subquestions, it was decided to create another section of the survey dedicated to it.

5.4 Expected Statistical Results

Expected results form an important part of any survey construction. These results were identified during the application of the VFT method:

- Correlation between time spent formally studying, working and teaching computer programming and the languages with most affinity;
- Correlation between the most common technical characteristics of the languages and the affinity level;
- Preferred structural, syntatic and semantic programming languages characteristics;
- Correlation of personal background and affinity;
- Changes in favorite languages and the reasons for the new choice;
- Languages that most frequently lost or gained affinity after a change;
- Frequency at which the first learnt language presents high level of affinity;
- Correlation between career prospection and affinity to the languages with higher market share;
- Correlation between learning style and language affinity;
- Correlation between popularity and language affinity.

With this part done, the Value-Focused Thinking method successfuly helped the construction of a *Beta version* for the desired survey, that aimed at gathering feedback for improvements and validation. This version implemented the result of the VFT method almost entirely without changes, with the exception of characteristics that proved to be unbalanced in practice, such as the number of sections that raised from three to four in order to better separate the types of questions. Finally, feedback questions were added to the end of each section to gather opinions about the questionnaire *per se*. This proved to be of great value for assessing the questionnaire's main values and validate the process of Value-Focused Thinking.

[4] This decision was taken based on an opportunity, as explained in Subsection 4.3, and it might even be relevant in subsequent studies.

5.5 Feedback and Validation

The Beta version[5] of the survey was applied to three different groups of students from Universidade do Minho and Instituto Politécnico de Bragança, in order to test and validate the current implementation. Feedback from the respondents pointed towards a few notes:

- The question about household income were considered too intrusive by a few respondents;
- Some respondents had trouble understanding a few english terms in the questions, which in turn, made the survey harder to answer;
- The grid of possible answers to the questions about syntactic and semantic characeristics confused a few respondents. Some students initially considered that affinity choices were mutually exclusive, allowing only one answer of either "I don't like it", "I like it", or "I prefer it" for each snippet of code. In actuality, each answer could be selected for more than one snippet;
- Although the feedback was essentially positive, both in comprehension and duration of the survey, some notes stating that there were too many questions were collected. This was one of the main concerns about the survey and its values.

With this feedback in mind, the final version of the survey was prepared and will be published for open access in the near future. The changes that will be applied will not be translated back to the Value-Focused Thinking document, since it will be considered a snapshot of the planning process before the first round of feedback. If, otherwise, it is intended to be a live document, changes to the survey should be transcribed back.

6 Conclusion

Strategies to support computer programming education are numerous but still face several and interesting challenges. While many and diverse characteristics have been shown as influential to the teaching-learning process, affinity to a programming language as an influential factor is largely unexplored and might have a positive – or even negative – influence on the whole process.

In order to further understand this topic, a new survey has been constructed, concentrating its focus into realizing which characteristics (personal, technical, contextual etc.) influence affinity between programmers of any level and programming languages. Being a complex and in-depth approach to the continuation of this research, this survey was prepared in a formal manner, using the Value-Focused Thinking method to guide the whole process. This method lead to the definition of the survey's elements based on core values and its derived objectives, creating a highly focused Beta version. The final version of the new survey is now finished – taking into account feedback already gathered – and it is currently open for answers[6]. That version of the survey will be disseminated, in the near future, as much as possible to a broad community of students, teachers and practitioners of programming in order to collect a huge amount of answers; then the collected data will be statistically analyzed and the results will be published.

[5] A copy may be found in the following address: `https://bit.ly/3DDJFtv`.
[6] The survey may be found and fulfilled at the following address: `https://bit.ly/3MdDgIH`.

References

1. M. V. P. Almeida, L. M. Alves, M. J. V. Pereira, and G. A. R. Barbosa. EasyCoding - Methodology to Support Programming Learning. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASIcs)*, pages 1:1–1:8, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ICPEC.2020.1.
2. M.V.P. Almeida. Easycoding: Methodology to support programming learning. Master's thesis, Instituto Politécnico de Bragança, 2020.
3. A.G. Applin. Second language acquisition and cs1. *SIGCSE Bull.*, 33(1):174–178, February 2001. doi:10.1145/366413.364579.
4. D.R. Barbosa. Adequacy Analysis of Learning Resources in Adult Education. Master's thesis, Minho University, Braga, Portugal, October 2021.
5. Council of Europe. *Common European Framework of Reference for Languages: Learning, teaching, assessment – Companion volume*. Council of Europe Publishing, Strasbourg, France, 2020. URL: https://www.coe.int/lang-cefr.
6. R.R. Fenichel, J. Weizenbaum, and J.C. Yochelson. A program to teach programming. *Communications of the ACM*, 13(3):141–146, March 1970. doi:10.1145/362052.362053.
7. J. Figueiredo and F.J. García-Peñalvo. Building skills in introductory programming. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM'18, pages 46–50, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3284179.3284190.
8. P. Freire. *Pedagogia da Autonomia: Saberes necessários à prática educativa*. Paz e Terra, 2011.
9. A. Gomes and A.J. Mendes. Learning to program: difficulties and solutions. In *Proceedings of the 2007 ICEE International Conference on Engineering and Education*, ICEE '07. International Network on Engineering Education and Research, 2007.
10. P.J. Guo. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 1–14, New York, NY, USA, 2018. Association for Computing Machinery.
11. Ralph L. Keeney. Value-focused thinking: Identifying decision opportunities and creating alternatives. *European Journal of Operational Research*, 92(3):537–549, 1996. doi:10.1016/0377-2217(96)00004-5.
12. J. Piaget, M. Piercy, and D.E. Berlyne. *The Psychology of Intelligence*. Routledge classics. Routledge, 2001.
13. Redacted. Redacted for blind review purposes. In *Redacted for blind review purposes*, Redacted.
14. S.A. Robertson and M.P. Lee. The application of second natural language acquisition pedagogy to the teaching of programming languages—a research agenda. *SIGCSE Bulletin*, 27(4):9–12, December 1995. doi:10.1145/216511.216517.
15. Stack Overflow. Stack overflow developer survey, 2021. URL: https://insights.stackoverflow.com/survey/2021.
16. StatisticsTimes.com. Top computer languages, 2020. URL: http://statisticstimes.com/tech/top-computer-languages.php.
17. L.S. Vygotsky, E. Hanfmann, G. Vakar, and A. Kozulin. *Thought and Language*. The MIT Press. MIT Press, 2012.
18. B.C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: A study of twelve factors. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 184–188, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/364447.364581.