A Natural Language for Bulgarian Primary and Secondary Education

Iglika Nikolova-Stoupak
Staffordshire University
iglika.nikolova.stoupak@gmail.com

Abstract

This paper examines the qualities and applicability of a provisional programming language, especially designed for use by beginner-level students in Bulgarian primary and secondary schools. The necessity for such a language is investigated. Then, relevant features are defined, as inspired by various programming languages (notably, languages used in education and characterised with non-English syntax) and by general trends related to the achievement of natural language in software development. A survey is conducted to test young students' interaction with the language, and the latter's advantages and limitations are listed and discussed.

Keywords: natural-language-programming, elementary education, Bulgarian education, cultural and social implications

1. Context

1.1. Computer Education in Bulgaria

Generally speaking, Bulgarian people take pride in their country's relationship with computer science. They are ready to point out that John Atanasoff, the inventor of the first electronic digital computer, was of Bulgarian origin. In terms of contemporary context, many are happy to discover that Bulgaria's Internet speed and accessibility rank within the world's top lists ("Bulgaria Ranks World's 20th in Internet Speed, Accessibility", 2014).

Computer studies were first introduced in Bulgarian education as early as 1959, in the University of Sofia and under the course name "Computational Mathematics and Cybernetics" (Kaltinska, 2018). The then utilised Romanian ECM CIFA machine was replaced by the Bulgarian computer Vitosha in 1963 (2018). They both worked solely with machine code. Computer Science or Informatics entered specialised high school education in 1972 for grade levels 10 and 11 (i.e. student age 16-18) (2018). The subject became mandatory for Bulgarian education in 1986, starting from grade 6 (age 12-13), and the language Logo was most frequently used during the next few years (Zamfirov, 2016: 47).

In 2008, a survey was carried out among high school teachers of Mathematics and Science, revealing a common opinion that computer studies were integrated too late within the curriculum, especially given students' experience with technology as part of contemporary everyday life ("Popitahme uchitelite!":2). In order to fill this hole, and partly motivated by recent growth in the IT sector in the country, computer education is currently being redefined and popularized (Stoyanova, 2013). Thematic extracurricular activities are increasingly on offer for young children, such as via established institutions,

newly opened internationally affiliated organisms and subscription-based websites. Many of these courses invite even preschoolers, who are not expected to be able to read or write.

Public education has been attempting to keep up with the described trend. Over the past decade, various Bulgarian schools have participated in facultative initiatives related to computer studies (and associated teacher training) which have been highly successful (Ayvaz, 2018). “It is a fact that the driving force behind the maintenance of interest and the improvement of instruction is the entirely personal initiative of teachers and students,” “Popitahme uchitelite!” asserts (2008: 2). As of the academic year 2018/2019, a major development occurred with the subject becoming obligatory in the context of primary education (namely, grade 3), under the title “Computer Modelling”. The current plan is to gradually include this subject in the curriculum for grade 1 (Dyulgerova, 2018). The basic topics discussed within the course include: what a computer is and basic hardware components; the generation of user profiles; risks and precautions; and basic algorithms.

It does not come as a surprise that the described young computer curriculum is still largely imperfect. Firstly, teacher training takes place in solely a single day and specialises strictly in the material covered by the course (Regional Educational Management–Sofia, 2018), thus generating an issue in terms of teaching competence. Computer equipment is, unfortunately, extremely scarce. As of 2013, the country’s average is one computer for eleven students as opposed to a European average of one to five (Nikolov, 2013: 10). Negative consequences of the problem are already perceivable, as whilst good theoretical knowledge is objectively demonstrated by Bulgarian IT students, their practical skills remain far from satisfactory (10). Finally and very importantly, no programming language has been established that reflects the Bulgarian cultural and educational context. The international visual language Scratch is most commonly used; specifically, in its imperfect and partial Bulgarian translation.

Given the described gaps within the Computer Studies curriculum as present within Bulgarian education, it is relevant to undergo the current project. Especially following the unexpected and unprecedented necessity for online education during the academic year 2019/2020, computer education and computer literacy within the entire school curriculum have come to acquire a key role. Consequently, the lack of a defined, optimally usable and appealing programming language is to be becoming increasingly obvious.

1.2. International Trends

An important general trend in relation to contemporary programming languages is the quest for “natural” language; in other words, a programming language aims to be as close as possible to the programmer’s “human” language and, consequently, as removed as possible from the machine code that hides behind the offered interface. Examples of so-called “high-level” or natural languages include COBOL, Pegasus and Jaa (a Java dialect). Contemporary research is highly centered on similar languages’ development and optimisation, and they are especially praised when child or beginner programmers are concerned. Stefik and Siebert show through an experiment that users, notably inexperienced ones, find established languages like Python highly more intuitive to use than a made-up language named “Randomo”, whose commands are not designed to resemble human language (2013). Myers et al, stating as their goal “to make it possible for people to express their ideas in the same way they think about them”, conduct a detailed study with non-programmers to define and test the language and environment Human-centered Advances for the Novice Development of Software (HANDS), where animations and simulations are utilised to express meaning (2004: 47).

To go further, researchers tend to agree that the naturalness and intuitiveness of a programming language meant for beginners can benefit greatly if native rather than English-based syntax is used where applicable. For instance, according to the designer of Russian educational language KuMir, Dr. Leonov, it was mandatory that the language have native syntax, as introducing a foreign one would add to the already unavoidable initial confusion that students experience (2013: 137). On a similar note, Baron et al claim in relation to French-syntax educational language LSE that the use of French can both facilitate the learning

process and avoid the potential negative interference that a non-native language might have on learners (1985: 10).

Let us also note that upon reaching a particular target audience, software needs to be both translated and "culturalised." It is therefore important that national programming languages, whether adapted or specially developed, take into account the very culture at hand. An example of optimal development in this regard is Japanese educational language "Kotodama on Squeak", which not only uses local syntax but also seeks to appeal to an audience that values literary and esthetic language, such as by removing abbreviations and ensuring that sentences feature correct and varied grammar.

The last feature specific to educational languages that is going to be underlined is their involvement in a particular school curriculum as opposed to isolated use. The skills developed in a computer course can naturally be applied to other aspects of academic life, including general research, the completion of homework, and the assimilation of mathematical skills and notions. French-language LSE illustrates this interdisciplinary quality especially vividly with the large associated library of educational software made available by the French National Center for Educational Documentation, which was launched soon after the language's popularisation in the 1980s and covered all school disciplines (Baudé, 2016: 48). Interestingly, this plurality would not be unprecedented within Bulgarian education itself. In 1984, a Computer Science textbook by Nikolov and Sendova claimed to be instructive simultaneously in Logo programming, Mathematics, English and Russian (Zamfirov, 2016: 48). It would thus be relevant to point out that the previously emphasised importance of native language programming is not to say that English should be avoided per se, especially taking into consideration its major role in school curriculums and the contemporary global world in general; rather, it is the *availability* of one's native language that should be a key concern.

2. Research Design

2.1. Methods

Secondary research for the current project encompasses sources related to the educational and cultural context at hand as well as to applicable international practices (particularly focusing on educational programming languages with national syntaxes) and to global trends concerning natural language programming. Primary research analyses the development and application of a programming language as designed for use within primary and secondary education in Bulgaria. The provisional language, Monoglossia (1Gl), is tested by way of a survey issued among students, and its discussion aims at constructive conclusions pointing at further work. Taking into account the importance of potential use of the language even without the presence of a computer (as mandated by current restrictions of equipment in Bulgarian public classrooms), the survey was printed out and completed in pen. For the language's full Syntax specification, please refer to Appendix D (Nikolova-Stoupak, 2020).

2.2. Participants

The survey was distributed to 20 children from a Bulgarian primary school, aged from 9 to 12. Their prior experience with programming was minimal, and their level of English was basic (with the exception of one child, bilingual in English and Bulgarian).

3. Primary Research

3.1. Data Collection: Survey

The survey (Nikolova-Stoupak, 2020 Appendix A) was completed by 20 children from a Bulgarian primary school, aged 9-12. They either had no formal experience with programming or had been following a Computer Basics course at school for no more than a few months. With the exception of one bilingual child, the participants' knowledge of English was beginner.

The survey consists of three programming exercises. Exercise 1 involves writing code in Bulgarian in the language prototype 1Gl, exercise 2 requires coding in English in 1Gl, and exercise 3 is composed in Logo, the English-based educational language that has historically been used in Bulgarian education. No

prior knowledge is required on the side of the students, as there are explanations and examples of all utilised constructions. Each exercise includes basic visual commands in its first part and the use of a simple loop in its second part.

The students were asked to record the time it took them to complete each exercise (in minutes). The exercises were followed by two "yes/no" questions: whether it was easier to program in Bulgarian; and whether writing code in English feels like English language practice to them. Finally, they were asked to identify the exercise that they deemed most difficult.

Via quantitative analysis, the survey seeks to achieve the following goals:

- Examine the kinds of language-related mistakes committed by students and their occurrence.
- Compare the number of non-language-related mistakes in 1Gl and Logo.
- Determine whether programming in Bulgarian was easier for the students.
- Correlate the timing of completion of the three exercises.

For all raw data involved in the discussion, please refer to Appendix B (Nikolova-Stoupak, 2020).

3.2. Results and Data Analysis

- Figure 1 presents the types of language-related mistakes committed by students along with percentage. Almost half of all mistakes are related to spelling (46.5 %), mistakes involving non-existent syntax following at 38.6%. Mistakes in punctuation and other undefined mistakes come together at 14.8%.

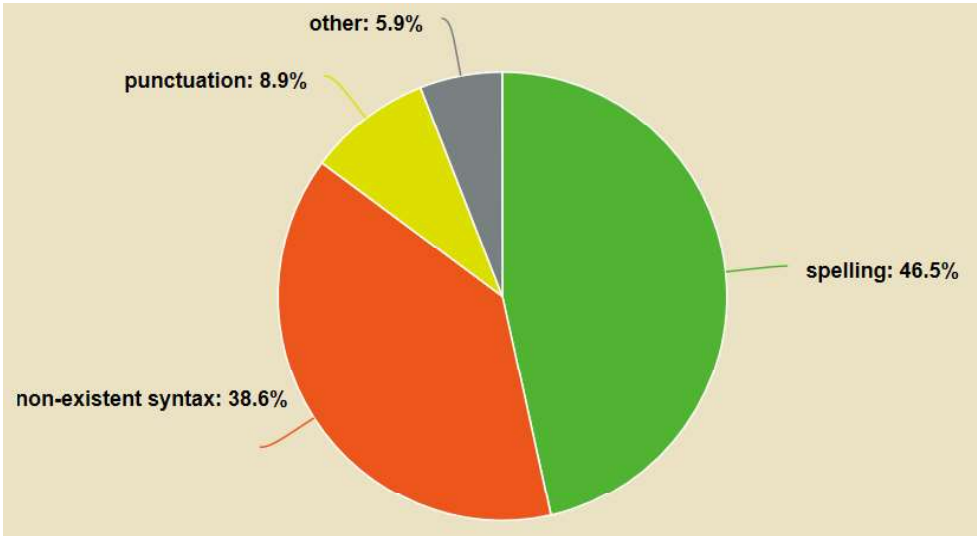

Figure 1: Types of language-related mistakes committed by students

It is important to note that, as should be expected, the vast majority of language-related mistakes (81%) were committed in the more natural-language-like programming language, 1Gl. The presence of spelling mistakes (especially in English) is explainable given the young age of students and the fact that most of them completed the survey on paper and without access to language tools. Mistakes linked to syntax were mostly based on wrong assumptions about the programming language's syntax (such as the writing of non-existing commands), inspired by students' experience with human language.

- As Figure 2 shows, the number of non-language-related mistakes (including use of symbols, use of loops, missing commands, unnecessary commands, misunderstood instructions, wrong calculation and non-optimal programming practice) is significantly higher in the Logo exercise (68) as compared with 1Gl (50 for Bulgarian-based and 55 for English-based). It may thus be suggested that the natural quality achieved by 1Gl aids in the prevention of mistakes of mathematical and logical nature.

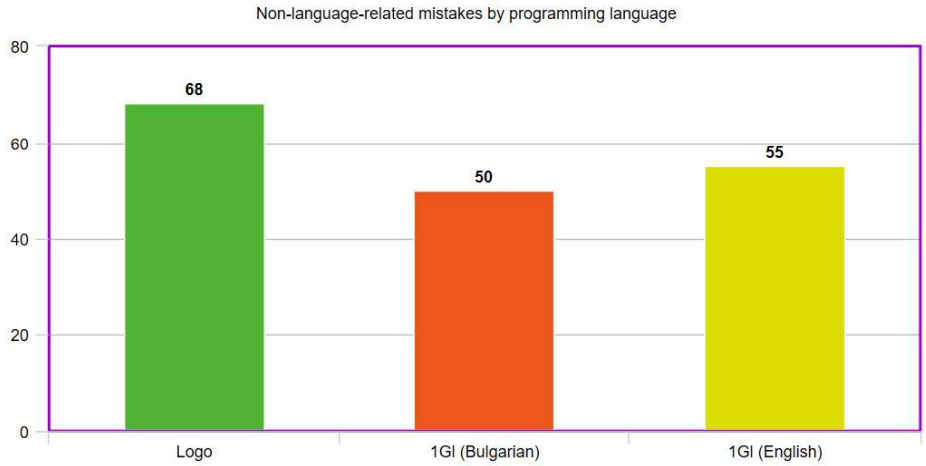

Figure 2: Number of non-language-related mistakes by programming language

- When asked about the most difficult exercise in their judgment, 7 students selected exercise 1, 8 students - exercise 2, and 5 students - exercise 3. Whilst this distribution is too even to welcome generalisations, one may consider the possibility that the fact that English-based 1Gl was voted as most difficult by one student more than Bulgarian-based 1Gl, even though the English-language exercise had the privilege of being very similar to the previous one, implies that Bulgarian-syntax programming tends to be perceived as easier.

The explicit question of whether students found programming in Bulgarian easier than programming in English was met with 16 (or 80% of) positive answers. This result further supports the previously presented suggestion. Yet, possible external influences behind answers should be noted; for instance, that "yes" is generally a more ready answer than "no" for young students and that the word "Bulgarian" may by default be associated with a feeling of ease as compared with "English". To further test students' honesty and validity of judgment at answering the question, cross tabulation was performed, associating the answers provided with the number of language-based mistakes actually committed in the two versions of 1Gl (see Table 1).

	Higher No. of Mistakes			
Answer	**Bulgarian**	**English**	**Equal**	**Total**
No	2	2	0	4
Yes	3	11	2	16
Total	5	13	2	20

Table 1: Cross tabulation between the number of language-based mistakes in the two versions of 1Gl and students' answers to the question of whether they found programming in Bulgarian easier.

Only 19 % of respondents who answered in the affirmative made a higher number of linguistic mistakes in Bulgarian, pointing to general validity of the answers. Also, 50% of respondents who answered

"no" actually made more mistakes in Bulgarian (despite the overall lower number of mistakes in the language), showing that judgment was mostly adequate. In fact, these two students make up for a whole 40% of all respondents who made more mistakes in Bulgarian.

When asked the next question i.e. whether they deemed that programming in English could enhance their knowledge in the language, the majority of respondents (75%) answered positively. As thorough analysis of the truthfulness of this statement can only be achieved through a detailed temporal study, an assumption will need to be made that the general correctness of students' judgment in relation to the previous questions is also applicable here.

- Table 2 shows that only two students were faster in the English version of 1Gl than in the Bulgarian one. Notably, one of them was the single bilingual child (this result is explainable through the fact that English syntax is significantly shorter and, therefore, naturally easier to use in the presence of identical knowledge of the two languages).

Student No.	Exercise 1	Exercise 2	Exercise 3
1	11	9	8
2	15	15	10
3	5	8	5
4	8	8	8
5	13	15	12
6	10	15	10
7	5	5	5
8	18	20	15
9	14	16	11
10	6	8	10
11	3	3	3
12	9	8	7
13	10	12	10
14	20	25	15
15	4	5	3
16	7	9	10
17	17	18	13
18	6	6	5
19	11	19	12
20	8	10	10

Table 2: Time (in minutes) of completion for each exercise, as provided by students

The Logo exercise took longest to complete for only 2 (10%) of the participants. It shared the smallest number of minutes with one or both of the other exercises in another 7 of cases, and took shortest to complete

for a whole 10 or 50% of cases. It seems therefore safe to assume that the language Logo is faster to compose code in than 1Gl. However, such a difference in timing is to be expected given that one of the main ideas behind 1Gl is, as explained, its multidisciplinary nature. In other words, at using 1Gl, a student may take longer than at working with a classical beginner programming language, but they are using this time to simultaneously build skills in an increased number of academic disciplines (notably, linguistics and ESL).

3.3. Discussion

Coding in 1Gl may introduce types of mistakes that are not readily committed in languages with less natural and/or non-native syntax; notably, spelling mistakes and wrong assumptions concerning syntax. The former could be reduced with the involvement of language editing software and even simply through continuous practice with the language, coincidently leading to an improvement of students' general spelling skills. Wrongly assumed syntax is more difficult to address, and it comes as an established problem in relation to natural programming languages. For instance, within his study of natural-language programming in English, Buckman states that "[i]n most cases, such errors involve a child guessing at a command's name or the syntax of its arguments" (1999: 211). The problem may be addressed through an efficient system of error messaging as well as additional instruction concerning computer logic and limitations (notably, computers' inability to understand unedited everyday speech). It is also important to note that, partly making up for the newly introduced language-related mistakes, more traditional ones (such as skipped symbols or commands) are likely to be reduced due to the language's intuitiveness.

As seen, students are likely to have an initial preference for coding in Bulgarian. This tendency works toward proving the highly supported opinion that it is psychologically and practically beneficial for students to be able to use their native language within the field. It can, however, be expected that 1Gl's English version is to become increasingly more intuitive following regular practice with and association of the two syntax varieties.

The survey results show that the general time for the composition of code is higher in 1Gl than in a classical programming language. As mentioned, the time "lost" can be viewed as largely made up for, given the undelined expected acquisition of interdisciplinary skills during the programming process. Also, in the context of beginner computer studies, students are not to be encouraged to focus on their speed but to firstly ensure that code is correct and optimally structured.

3.4. Sources of Error and Suggestions for Further Work

Very importantly, not all major features of the proposed language, 1Gl, have been tested by the current survey. The main focus of the study is on natural language programming and multilingualism, and many syntax elements are left unaccounted for in order for exercises to remain simple enough for students with no programming background.

Another limitation of the study comes in the face of difficulties to differentiate between mistake types. For instance, if a student fails to colour the path taken by the actor, does this imply missing commands or a prior misunderstanding of instructions?

The study would be enriched by future involvement of a larger sample of respondents as well as by a wider age range (which would in turn imply more varied ESL skills, for instance). In the presence of a greater number of respondents, more elaborate statistical analyses such as a chi-square test would be applicable in the evaluation of relationships.

Finally, it should be noted that the Logo task has been perceived as rather simple by participants as compared with the other tasks. While this simplicity is not mainly due to the programming language in use (but to, for instance, similarity to the other two tasks and shorter commands), erroneous assumptions could be made in this direction. The selection of a slightly more complex task or of one slightly larger in size (for instance, the drawing of several adjacent figures instead of a single one) could improve the survey's accuracy.

4. Conclusion

Students are the main target group of users to rely on natural, human-like programming as well as to utlilise native (non-English) syntax. In particular, Bulgarian national education in its extending and increasingly early focus on computer studies is in need of unified and systematised programming practice. This project evaluated the potential benefits of a proposed programming language for beginner students with dual Bulgarian and English syntax. Ideally, the study will proceed to further development of the language's characteristics, accompanied with gradual analyses of its reception by Bulgarian students.

Ethical Consideration

Ethical approval has been granted by the associated higher education institution prior to the project's completion. All participants in the utilised survey are anonymous, and parents have agreed to the participation of their children in the project.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Ayvaz, H. (2018). Kompyutarno Modelirane v Uchilishte za Po-razchupeno Obrazovanie, *Bloomerang TV*. https://www.bloombergtv.bg/biznes-start/2018-02-21/kompyutarno-modelirane-v-uchilishte-za-po-razchupeno-obrazovanie.

Baron, G. et al (1985). Dix ans d'informatique dans l'enseignement secondaire : 1970-1980, *Institut national de recherche pédagogique.* http://lara.inist.fr/bitstream/handle/2332/1250/INRP_RP_81_113op.pdf?sequence=2.

Baudé, J. (2016). 'Le système LSE.' *EpiNet* (182): 41-56. http://www.societe-informatique-de-france.fr/wp-content/uploads/2015/12/1024-no7-Baude.pdf.

Bruckman, A. and Edwards, K. (1999). Should We Leverage Natural-Language Knowledge? An Analysis of User Errors in a Natural-Language-Style Programming Language*, Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, USA, 15-20 May, pp. 207-214. https://doi.org/10.1145/302979.303040.

Bulgaria Ranks World's 20th in Internet Speed, Accessibility (2014) *Sofia News Agency*. https://www.novinite.com/articles/164134/

Dyulgerova, D. (2018). 'Predmeti kato Kompyutarno Modelirane i Informatsionni Tehnologii shte Razvivat Digitalnoto Obrazovanie v Stranata', *Focus News Agency*. http://www.focus-news.net/news/0000/00/00/2574674/.

Kaltinska, R. Nachalo na Informatikata v Balgariya (1959-1980). *Bulgarian Museum of Mathematics and Computer Science*. http://mmib.math.bas.bg/?page_id=5612.

Leonov, A. G. (2013). The Logical Design of Pedagogical Programming Systems, *Yaroslavskiy Pedagogicheskiy Vestnik*, 3(4):134-141.

Milanova, A. N. et al (2018). *Kompyutarno Modelirane za 3 Klas*. Sofia: Prosveta Plus.

Popitahme uchitelite! (2008). *Infoman.bg*. infoman.musala.com/files/view/static/anniversary/articles/AskTeachers.pdf/.

Myers, B. A. et al (2004). Natural Programming Languages and Environments. *Communications of the ACM*, 47(9):47-52. https://doi.org/10.1145/1015864.1015888.

Nikolov, A. (2013) Uchilishtnoto obrazovanie v Balgariya: Sastoyanie i Tendentsii. *Institute for Market Economics*. https://ime.bg/var/images/secondary_education_Adrian.pdf.

Nikolova-Stoupak, I. (2020) Appendices. In "A Natural Language for Bulgarian Primary and Secondary Education". *Computational Linguistics in Bulgaria 2020*. Sofia, Bulgaria, 25-26 June. https://www.kaggle.com/iglikastoupak/natural-language-for-bulgarian-education?select=Appendices.docx.

Regional Educational Management - Sofia. (2018) *Obuchenie po Kompyutarno Modelirane za Uchiteli* [Press release]. 6 November. http://ruo-sofia-grad.com/новини/обучение-по-компютърно-моделиране-за-учители.

Stefik, A. and Siebert, S. (2013). An Emprical Investigation into Programming Langauge Syntax, *Transactions on Computing Education (TOCE)*, 13(4):1-40. https://doi.org/10.1145/2534973.

Stoyanova, S. (2013). Mahat informatikata ot chasovete v uchilishte? *Dnes.bg*. https://www.dnes.bg/obshtestvo/2013/12/13/mahat-informatikata-ot-chasovete-v-uchilishte.209454.

Zamfirov, M. (2016). Sitoricheski predpostavki za vnedryavaneto I razvivaneto na obuchenieto po informatika v balgarskite uchilishta. *3rd Congress of Physical Sciences*. University of Sofia, Sofia, Bulgaria, 29 September -2 October, pp 47-50. www.trioiskar.com/hp/2016-12mzamfirov.pdf.