"*Hey IDE, Display Hello World*": Integrating a Voice Coding Approach in Hands-on Computer Programming Activities

Manuel B. Garcia
FEU Institute of Technology
Manila, Philippines
mbgarcia@feutech.edu.ph

John Benedic R. Enriquez
FEU Institute of Technology
Manila, Philippines
jrenriquez@feutech.edu.ph

Rossana T. Adao
FEU Institute of Technology
Manila, Philippines
radao@fit.edu.ph

Ari Happonen
LUT University
Lappeenranta, Finland
happonen@lut.fi

2022 IEEE 14th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM) | 978-1-6654-6493-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/HNICEM57413.2022.10109412

***Abstract*—Following recent advancements in automatic speech recognition (ASR) technologies, we replicated an experiment four decades ago that utilized voice as an input modality for computer programming. We also extended this experiment by investigating the pedagogical effectiveness of '*programming by voice*' in terms of attitude, self-efficacy, code correctness, and coding speed. A total of 96 students from an institute of technology in the capital region of the Philippines were randomly selected to participate in a quasi-experimental study using a one-group pretest-posttest design. We subjected students to programming activities with different levels of difficulty to compare voice and keyboard. Our results show that although voice decreases negativity, it likewise decreases control, which means that both attitude and self-efficacy are positively and negatively affected, respectively. Using voice as an input modality also allows students to code faster when the activities are easy but not when they are moderate or difficult. Code correctness analysis shows that voice is only preferable for easy and moderate machine problems. With the deviation of our findings from an experiment four decades ago, we can now conclude that ASR technologies and voice as input modality provide substantial implications and new opportunities for teaching and learning computer programming.**

Keywords— Automatic Speech Recognition, Human-Computer Interaction, Computer Programming, Coding

I. Introduction

In the interdisciplinary subfield of computational linguistics and computer science, Automatic Speech Recognition (ASR) is defined as the automated process of converting voice input into its corresponding transcript. It is an important research domain in human-computer interaction (HCI) due to its vast real-world applications that unceasingly alter the way people live [1-3]. The implications of ASR technologies led to classifications based on how they strengthen human-human communication (HHC) and human-machine communication (HMC) [4]. One example is the obstacle brought by monolingualism in HHC, particularly when people communicate with people from another language group. ASR alleviates this difficulty and eliminates the language barrier through automated speech-to-speech translation (e.g., Asian and English languages [5]). In terms of HMC, there have been many exciting advancements and voice assistants (e.g., Siri, Cortana, and Alexa) have become mainstream technologies because they come inbuilt into numerous devices, including smartphones and computers. The ubiquitousness of ASR is most apparent in the persistent proliferation of voice assistants in many sectors, such as health [6-8], business [9-11], education [12-14], and others.

The reputation of computer programming as an intellectually challenging course has conceived a new phenomenon known as *programming anxiety*. Sometimes referred to as *fear of coding*, many researchers have studied the reasons for its occurrence and prospective solutions due to its negative effects on the learning process. It has been posited that this psychological state occurs because students mistakenly assess their programming learning ability compounded by the underdevelopment of requisite skills [15]. With the shortage of self-efficacy and a sense of control, it is consequently crucial to formulate strategies that foster student motivation and confidence. Meanwhile, fear was examined as a descriptor that symbolizes a lack of appreciation of or interest in computer programming as a discipline [16]. Accordingly, there is a prevalence of feelings of discomfort or apprehension among tertiary students. In addition to internal factors (e.g., attitude and motivation), external factors (e.g., teachers and their strategies) were also emphasized as catalysts for the inhibiting force of fear. In an attempt to reduce, if not eliminate, programming anxiety, different strategies have been proposed, such as utilizing online interactive coding platforms [17], gamified online courses [18], technology-supported interactive strategies [19], group learning approaches [20], educational programming languages [21], and others. A common denominator among these pedagogies is the provision for a more engaging and active learning experience.

Almost four decades ago, one study conducted a controlled experiment to compare voice and keyboard as input modalities in writing computer programs [22]. The basis of this study leans on the promising enrichments offered by voice inputs in crafting effective user interfaces that minimize keyboard operations and maximize real-time naturalistic HCI. Through the measures of accuracy, speed, and efficiency, the experiment discovered that voice competes reasonably well with the keyboard. Nonetheless, it was hypothesized further that voice could have outperformed the keyboard in all aspects had the computer processing power been sufficient and participants were more experienced in voice-enabled devices. This assumption warrants further investigation. Following recent advancements in ASR technologies, this study replicates the experiment and likewise extends the evaluation by assessing the pedagogical effectiveness of a voice-driven coding approach. Understanding the effectiveness of voice inputs using ASR technologies may present new and exciting opportunities for teaching and learning computer programming. Accordingly, the findings of this experiment will benefit teachers and students of computer programming in their intended academic outcomes.

In this study, we explicitly referred to the use of voice instead of a keyboard to write source code as *voice programming*. This *programming by voice* approach is a relatively underdeveloped research area and deserves further investigation in light of recent developments in ASR technologies and HCI concepts. There are also significant health concerns about computer-related injuries caused by excessive typing. In addition, some motor disabilities prevent computer users from using a keyboard and mouse. Thus, the significance of our study is not constrained to programming pedagogies but also implications that extend to health disability inclusion. These research questions (RQs) guided our study:

RQ1. Is there a significant difference in terms of attitude and self-efficacy before and after the activities?

RQ2. Is there a significant difference in the code correctness between voice and typed input?

RQ3. Is there a significant difference in the coding speed in terms of modality and difficulty of machine problems?

II. Background of the Study

A. Automatic Speech Recognition

For human beings, verbal communication is the most natural form of communication. Therefore, teaching computers to learn and understand natural human languages (e.g., natural language processing) is an unsurprising idea. Supported by existing voice and speech technologies, a naturalistic interaction between users and computers and other digital devices using verbal commands are no longer a fictional scenario [23]. This interaction is largely attributed to the continuous advancements in the field of speech signal processing, particularly in ASR technologies.

B. Voice User Interface

There has been a growing number of devices that integrate a voice-user interface (VUI) making ASR more convenient. VUI is a technology that provides ASR capabilities to assist users in interacting with their electronic devices using voice commands. Some applications include home automation systems [24], smart speakers [25], service robots [26], and others. In education, VUI can also support the teaching and learning process. For instance, schools can enhance their student support services through VUI-enabled applications. One example is the chat system that assists computer science students in their academic concerns, including referencing, academic writing, and programming [27].

C. Computer-Related Injuries

Computer-related injuries (e.g., carpal tunnel syndrome) are a prevalent health issue among computer users. Daily computer use of at least four hours significantly increased frequent health complaints, especially on hands, fingers, and wrists [28]. There is also extensive evidence that keyboard users are susceptible to Repetitive Strain Injury (also known as overuse syndrome) [29]. For computer programmers who are prolonged keyboard users, this health issue is a serious problem. Ergonomic keyboards are a solution that has been extensively used by typists [30]. Albeit this product is also applicable for coders, eliminating keyboards and replacing them with VUI devices is worth exploring. Some studies have already examined a hands-free computer interface approach in voice-assisted software modeling [31] and speech-based integrated programming environments [32, 33].

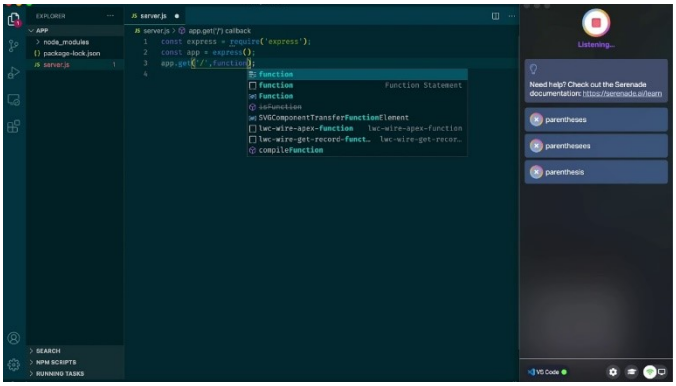

Fig. 1. Voice-Driven Integrated Development Environment Using Serenade.

D. Students with Disabilities

VUIs are also appealing to users who experience difficulties in the conventional graphical user interface [34]. For instance, a series of co-design workshops were conducted to develop VUIs with and for visually-impaired students [35]. The inspiration for this study was the proliferation of voice-based personal assistant devices. It was learned that VUIs possess a significant potential for creating accessible and inclusive interactions in an academic setting. Another proof supporting this claim is the utilization of a "*programming by voice*" approach for motorically challenged children [36-38]. The empirical findings from prior works have propelled ASR technologies and VUI devices into the education sector as a potential instructional technology intervention.

III. Methodology

A. Research Design

This cross-sectional experimental research with a one-group pretest-posttest design is an empirical investigation of ASR and voice programming. In recent years, ASR technologies through VUI devices have been catapulted into many sectors of society, making them a subject of interest among researchers. Instead of an exploratory nature of research like in previous investigations, we purposely selected an experimental methodology to examine cause-effect relationships. Driven by the same notion that voice is the most natural form of human communication, we replicated a controlled experiment conducted almost four decades ago [22]. This experiment evaluated voice versus keyboard as modalities for writing computer programs. We extended this experiment by assessing the efficacy of voice programming in terms of attitude, self-efficacy, code correctness, and coding speed.

B. Setting and Sample

Students from an institute of technology in the capital region of the Philippines were randomly selected to join the study. This non-sectarian, private higher educational institution offers four-year information technology and computer science programs. At the time of our research, the Bachelor of Science in Information Technology (BSIT) has four specializations, such as Animation and Game Development (BSIT-AGD), Digital Arts (BSIT-DA), Business Analytics and/or Service Management (BSIT-SMBA), and Web and Mobile Applications (BSIT-WMA). Regardless of multiple specializations, the BSIT degree program positions the computer programming courses as a potent foundation not only for academic development but also for a future computing career

and employability prospects. This standpoint is noticeable in the numerous programming courses in the curriculums. The sample size was computed using Slovin's formula $n = N \div (1+Ne2)$ and the inclusion criteria were (1) a passing grade in the introductory programming course and (2) enrollment in any of the subsequent courses (e.g., Object-Oriented Programming). With a population of 126 programming students, the preferred sample size was 96. All students agreed and submitted an informed consent form.

C. Measurement and Data Collection

In our data collection, we utilized a survey questionnaire that has been similarly used in another programming study [20]. This questionnaire contains demographic information and measures of programming attitude and self-efficacy using validated scales namely the *Attitude Scale of Computer Programming Learning* (ASCOPL) and the *Computer Programming Self-Efficacy Scale* (CPSES), respectively. The demographic information section is composed of students' age, gender, program specialization, and prior programming grade. The ASCOPL is a five-point Likert with three constructs (i.e., willingness, negativity, and necessity) measuring attitude toward learning computer programming. The CPSES is an evaluation tool with five constructs (i.e., algorithm, logical thinking, debug, control, and cooperation) that measure students' beliefs in their capability to perform well in computer programming courses. The questionnaire was distributed before and after the experiment for the pretest and posttest comparison. The experiment was comprised of programming activities with three difficulty levels. The easy level includes Input and Output, Arithmetic Operation, and Variable Manipulation. The average level includes Conditional Statements, Looping Structures, and Standard Library Functions. Finally, the difficult level includes Array Data Structures and User-Defined Functions. Some of the programming activities we used for each level are as follows:

Easy: *Write a program that will prompt users to input their name, section, and other custom information about them as well as class schedules (at least five courses with complete details such as the time, room, teacher, etc.) After students encoded all their details, clear the whole screen and then display the output in an organized, COR-inspired layout.*

Moderate: *Write a program that reads a date from the user and computes its immediate successor. For example, if the user enters values that represent 2022-11-18 then your program should display a message indicating that the day immediately after 2022-11-18 is 2022-11-19. If the user enters values that represent 2022-11-30 then the program should indicate that the next day is 2022-12-01. If the user enters values that represent 2022-12-31 then the program should indicate that the next day is 2023-01-01. The date will be entered in numeric form with three separate input statements: one for the year, one for the month, and one for the day. Never mind the leap year for now.*

Difficult: *Write a program that can simulate an ATM. In this program, you should be able to integrate different programming techniques such as conditional statements, looping constructs, and functions. It is also important to use a multidimensional array for the card and PIN (login).*

D. Data Analysis

The collected data were analyzed using IBM SPSS Statistics 26.0. We utilized descriptive statistics to report the demographic information. Since ordinal data were produced by the ASCOPL and CPSES instruments, we used the Wilcoxon signed-rank test for the comparison of pretest and posttest scores of attitude and self-efficacy (RQ1). Rather than the efficiency of the algorithm, we graded the programming activities based on code correctness (i.e., code is running and achieves the correct output) making it a dichotomous variable. Thus, McNemar's test was utilized via a cross-over design where programming students write solutions to machine problems using both keyboard and voice (RQ2). For RQ3, we employed MANOVA to determine whether the coding speed varies in terms of difficulty levels of activities. All levels have ten pre-made activities and each activity was written in three different formats ($n = 30$) for randomization purposes.

IV. Results

A. Demographic Profile

Table 1 presents the profile of the respondents. A total of 96 programming students joined the experimental study. The mean age was 18.78 ± 0.81 years, and the majority were male students enrolled in a BSIT-WMA program. In the earlier programming course, their average grade ranged from 81-85% (mean = 83%).

TABLE I. Demographic Profile of Programming Students

Profile	Classification	f	%
Age	Less than 18 years old	13	18.75
	18 years old and above	83	81.25
Gender	Male	66	68.75
	Female	30	31.25
Specialization	BSIT-DA	13	13.54
	BSIT-WMA	39	40.63
	BSIT-AGD	25	26.04
	BSIT-SMBA	19	19.79
Previous Programming Grade	70-75	7	7.29
	76-80	19	19.79
	81-85	37	38.54
	86-90	18	18.75
	91-95	10	10.42
	96-100	5	5.21

B. Attitude and Self-Efficacy

The evaluation of programming input modalities in terms of attitude and self-efficacy exhibited mixed findings according to Wilcoxon signed-rank test (Table 2). In terms of attitude, only negativity was the significant construct decreasing from 3.85 ± 1.44 to 3.55 ± 1.15 ($p = 0.174$). Although the scores decreased, it is still a positive result as it implies that negative perceptions of computer programming were reduced after the experiment. Both willingness (3.11 ± 1.50 to 3.49 ± 1.15) and necessity (from 3.22 ± 1.42 to 3.40 ± 1.18) increased but not significantly ($p = 1.000$). When it comes to self-efficacy, control was the only significant construct decreasing from 4.01 ± 0.81 to 3.36 ± 1.12 ($p = 0.11$). Unfortunately, this finding indicates that students were more in control of using the keyboard than voice inputs.

TABLE II. ATTITUDE AND SELF-EFFICACY SCORES

Factor	Constructs	Pretest ($M \pm SD$)	Posttest ($M \pm SD$)	p-value
Attitude	Willingness	3.11 ± 1.50	3.49 ± 1.15	1.000
	Negativity	3.85 ± 1.44	3.35 ± 1.15	0.044
	Necessity	3.22 ± 1.42	3.40 ± 1.18	1.000
Self-Efficacy	Logical Thinking	3.43 ± 1.50	3.10 ± 1.40	1.000
	Algorithm	2.53 ± 1.17	2.69 ± 1.15	1.000
	Debug	3.58 ± 1.11	3.98 ± 0.85	0.823
	Control	4.01 ± 0.81	3.36 ± 1.12	0.011
	Cooperation	3.65 ± 1.16	4.05 ± 0.83	1.000

C. Code Correctness

The evaluation of programming input modalities in terms of code correctness likewise revealed mixed findings according to McNemar's test. Both easy ($p = 0.031$) and moderate ($p = 0.008$) activities yielded significant changes when students switched to voice as the input modality. For the difficult level, switching the modalities did not prompt significant changes ($p = 0.250$). In the crosstabulation (Table 3), it was shown that 29 students initially got incorrect solutions but correctly answered the easy problems after switching to voice programming. The same results can be observed in average problems, where 20 students originally got incorrect answers but made correct solutions after using voice as the input modality. Finally, there were more negative changes in the difficult level (correct to incorrect) albeit not significantly.

TABLE III. CODE CORRECTNESS USING KEYBOARD AND VOICE

Difficulty Levels	Keyboard	Voice		*p*-value
		Incorrect	Correct	
Level 1 – Easy	Incorrect	9	29	0.031
	Correct	6	52	
	Total	15	81	
Level 2 – Average	Incorrect	15	20	0.008
	Correct	8	53	
	Total	23	73	
Level 3 – Difficult	Incorrect	15	8	0.250
	Correct	13	60	
	Total	28	68	

D. Coding Speed

According to the one-way MANOVA analysis (Table 4), the coding speed between keyboard and voice as the input modality was significant in terms of difficulty levels ($p = 0.039$; Wilk's $\Lambda = 0.542$, partial $\eta^2 = 0.434$). Interestingly, voice is faster (167.03 ± 81.06 seconds) than keyboard (173.30 ± 76.45 seconds) when the difficulty level of the machine problem is easy. On the other hand, students finish their moderate and difficult activities faster when they use a keyboard (538.76 ± 249.16 and 886.80 ± 172.24 seconds) instead of the voice programming technique (545.07 ± 253.09 and 900.80 ± 176.68 seconds). This finding indicates that voice programming may only be useful for easy activities.

TABLE IV. COMPARISON OF CODING SPEED

Coding Speed	Keyboard	Voice
Level 1 – Easy		
M ± SD	173.30 ± 76.45	167.03 ± 81.06
Min – Max	43 – 292	30 – 294
Level 2 – Average		
M ± SD	538.76 ± 249.16	545.07 ± 253.09
Min – Max	107 – 994	112 – 997
Level 3 – Difficult		
M ± SD	886.80 ± 172.24	900.80 ± 176.68
Min – Max	608 – 1196	631 – 1314

V. DISCUSSION

In this experimental study, we examined voice programming as a coding approach in terms of attitude, self-efficacy, coding speed, and code correctness among programming students. This is a replication and extension of an experiment on using voice as an input modality in computer programming [22]. A total of 96 random students from an institute of technology in the capital region of the Philippines participated in a series of programming activities with three difficulty levels for both voice and keyboard modalities. Overall, our study showed mixed findings.

According to our findings, the negativity towards computer programming as a discipline significantly decreased after using the voice modality. Computer programming has a reputation for being a difficult course and many students are afraid of learning it because of this perception [20]. More importantly, students are more likely to believe computer programming is difficult when they have negative impressions of the subject. The fear factor is also detrimental because it diminishes intrinsic motivation and negatively influences student attitudes [16]. Therefore, the effect of voice programming opens positive opportunities for teachers to engage their students who have a negative attitude toward this subject. One possible explanation for the reduction of negativity could be attributed to the enjoyment when using voice inputs. In a coding workshop, it was found that fun programming activities have a positive effect on student attitudes towards coding [39].

Despite the positive effect of voice programming on attitude, the opposite is evident in the self-efficacy construct. According to students, using a voice interface significantly decreased their perceived control. This finding is consistent with what has been discovered in the original experiment [22], where less input task completion rate through the voice editor was recorded compared to the key editor. One possible explanation is that the keyboard has been the primary text input device since the introduction of computers and so people are more accustomed to it than to voice and speech recognition technologies. The positive and negative impact of employing a voice programming approach to attitude and self-efficacy, respectively, requires teachers to balance their implementation of this programming pedagogy. Importantly, it was found that perceived academic control is a crucial factor to predict dropout intention through the mediation of anxiety [40].

In terms of code correctness, it appears that voice modality benefits students only when solving easy and moderate machine problems. This finding could be attributed to the small cognitive load demanded by these activities, allowing students to enjoy the

voice interface more. However, when the activity is difficult and requires strict concentration, using a voice modality did not elicit significantly positive changes. Thus, it would be more beneficial to introduce the voice programming approach in the early stage of students' coding journey. This tactic is also compatible with the necessity to teach basic concepts first (e.g., syntax, variables, expressions, and operators) before complex algorithms [41].

The coding speed also varied depending on the difficulty of the programming activities, which is expected because easy ones require less time to finish and vice versa. However, it was clear that students were able to maximize the voice interface to finish easy activities but not moderate and difficult ones. This finding is similar to previous studies that found the speech input method was faster than keyboard input [42, 43]. However, it contradicts the original experiment [22], but the authors claimed that voice input as the mode of coding would have been more competitive with the keyboard had the technologies been sufficient. With the readiness of more advanced ASR technologies, it explains why voice inputs can be faster than keyboard inputs. This finding also verifies our result regarding code correctness rate, indicating the benefits of voice programming to easy machine problems.

Our results offer considerable implications for programming teachers and students given the widespread belief that computer programming is a challenging subject [15, 20]. With the positive influence of a voice programming approach on student attitudes, teachers may incorporate the voice interface into their laboratory activities. When students do not have a negative attitude toward the course, they are more likely to develop their self-efficacy and academic performance [39]. However, caution must be taken in terms of the timing of implementation. According to our results, the voice programming approach is only advantageous when the activities are easy and short. Thus, implementing it in activities that impose a higher cognitive load may result in negative effects academically. Finally, previous studies underscored the need for ASR technologies with adequate technical capabilities to ensure the success of a voice-driven programming approach [22, 42]. It emphasizes the necessity for schools to invest in equipment and training to ensure the success of its implementation.

There are limitations to our study that present future avenues for research. First, our study was limited to students enrolled in a computing degree, which indicates that we did not incorporate other non-computing degrees (e.g., engineering) that also offer computer programming courses. In coding-based events, it was highlighted to involve students from different programs as they may have different perceptions of the activity [44]. In terms of attitude and self-efficacy, the instrument was self-administered, which may result in bias and social desirability. The experiment was also reliant on the employment of voice technologies, which indicates that schools must modify their classroom and ensure a voice-enabled development environment and a microphone are available. It may also be challenging to have all students talking at the same time – a potential issue that we have avoided because of the mandatory online education during the pandemic. Finally, future researchers may consider replicating the experiment with the participation of programmers with different levels of ability. It is possible that more advanced programmers who are used to writing codes and able to formulate logic correctly may find the voice programming approach more beneficial.

VI. Conclusion

In this paper, we investigated the pedagogical potential of a voice programming approach concerning attitude, self-efficacy, code correctness, and coding speed. Our results demonstrate that although voice as an input modality decreases negativity, it also decreases control. This opposite effect reveals that both attitude and self-efficacy factors are positively and negatively affected, respectively, by the voice programming approach. Using a voice interface also allows students to code faster when the activities are easy but not when they are moderate or difficult. In our code correctness analysis, we found that utilizing voice input is only desirable for easy and moderate machine problems. Overall, our study upholds the pedagogical potential of utilizing voice as an input modality in writing computer programs. Considering these results, future researchers may explore the best way to integrate voice technologies either to replace or supplement keyboards.

References

[1] R. Hu, S. Zhu, J. Feng, and A. Sears, "Use of Speech Technology in Real Life Environment," in *Universal Access in Human-Computer Interaction. Applications and Services*, Berlin, Heidelberg, 2011: Springer, pp. 62-71, doi: 10.1007/978-3-642-21657-2_7.

[2] S.-O. Proksch, C. Wratil, and J. Wäckerle, "Testing the Validity of Automatic Speech Recognition for Political Text Analysis," *Political Analysis*, vol. 27, no. 3, pp. 339-359, 2019, doi: 10.1017/pan.2018.62.

[3] J. L. K. E. Fendji, D. C. M. Tala, B. O. Yenke, and M. Atemkeng, "Automatic Speech Recognition Using Limited Vocabulary: A Survey," *Applied Artificial Intelligence*, vol. 36, no. 1, pp. 1-35, 2022, doi: 10.1080/08839514.2022.2095039.

[4] D. Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach* (Signals and Communication Technology). London, UK: Springer, 2015, doi: 10.1007/978-1-4471-5779-3.

[5] S. Nakamura *et al*., "The ATR Multilingual Speech-to-Speech Translation System," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 2, pp. 365-376, 2006, doi: 10.1109/TSA.2005.860774.

[6] C. M. Seródio Figueiredo, T. de Melo, and R. Goes, "Evaluating Voice Assistants' Responses to COVID-19 Vaccination in Portuguese: Quality Assessment," *JMIR Human Factors*, vol. 9, no. 1, pp. 1-8, 2022, doi: 10.2196/34674.

[7] A. Schulte, R. Suarez-Ibarrola, D. Wegen, P.-F. Pohlmann, E. Petersen, and A. Miernik, "Automatic Speech Recognition in the Operating Room – An Essential Contemporary tool or a Redundant Gadget? A Survey Evaluation Among Physicians in Form of a Qualitative Study," *Annals of Medicine and Surgery*, vol. 59, pp. 81-85, 2020, doi: 10.1016/j.amsu.2020.09.015.

[8] A. S. Miner *et al*., "Assessing the Accuracy of Automatic Speech Recognition for Psychotherapy," *npj Digital Medicine*, vol. 3, no. 82, pp. 1-8, 2020, doi: 10.1038/s41746-020-0285-8.

[9] V. Rabassa, O. Sabri, and C. Spaletta, "Conversational Commerce: Do Biased Choices Offered by Voice Assistants' Technology Constrain its Appropriation?," *Technological Forecasting and Social Change*, vol. 174, pp. 1-12, 2022, doi: 10.1016/j.techfore.2021.121292.

[10] P. Hegdepatil and K. Davuluri, "Business Intelligence Based Novel Marketing Strategy Approach using Automatic Speech Recognition and Text Summarization," in *2021 2nd International Conference on Computing and Data Science (CDS)*, 2021, pp. 595-602, doi: 10.1109/CDS52072.2021.00108.

[11] H. M. Chang, "Is ASR Ready for Wireless Primetime: Measuring the Core Technology for Selected Applications," *Speech Communication*, vol. 31, no. 4, pp. 293-307, 2000, doi: 10.1016/S0167-6393(99)00063-1.

[12] J. H. Al Shamsi, M. Al-Emran, and K. Shaalan, "Understanding Key Drivers Affecting Students' Use of Artificial Intelligence-Based Voice Assistants," *Education and Information Technologies*, vol. 27, pp. 8071-8091, 2022, doi: 10.1007/s10639-022-10947-3.

[13] M. Russell *et al*., "Applications of Automatic Speech Recognition to Speech and Language Development in Young Children," in 4th International Conference on Spoken Language Processing (ICSLP), 1996, pp. 176-179, doi: 10.1109/ICSLP.1996.607069.

[14] B. Fox Carly, M. Israelsen-Augenstein, S. Jones, and L. Gillam Sandra, "An Evaluation of Expedited Transcription Methods for School-Age Children's Narrative Language: Automatic Speech Recognition and Real-Time Transcription," *Journal of Speech, Language, and Hearing Research*, vol. 64, no. 9, pp. 3533-3548, 2021/09/14 2021, doi: 10.1044/2021_JSLHR-21-00096.

[15] C. Connolly, E. Murphy, and S. Moore, "Programming Anxiety Amongst Computing Students—A Key in the Retention Debate?," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 52-56, 2009, doi: 10.1109/TE.2008.917193.

[16] 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, 2010, doi: 10.28945/1183.

[17] R. B. Figueroa Jr. and E. M. Amoloza, "Addressing Programming Anxiety among Non-Computer Science Distance Learners: A UPOU Case Study," *International Journal for Educational Media and Technology*, vol. 9, no. 1, pp. 56-67, 2015. Available: https://jaems.jp/contents/icomej/vol9/7_Figuroa.pdf.

[18] M. B. Garcia and T. F. Revano, "Assessing the Role of Python Programming Gamified Course on Students' Knowledge, Skills Performance, Attitude, and Self-Efficacy," in *2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2021, pp. 1-5, doi: 10.1109/HNICEM54116.2021.9731935.

[19] Y. Jiang, Z. Zhao, L. Wang, and S. Hu, "Research on the Influence of Technology-Enhanced Interactive Strategies on Programming Learning," in *2020 15th International Conference on Computer Science & Education (ICCSE)*, 2020, pp. 693-697, doi: 10.1109/ICCSE49874.2020.9201627.

[20] M. B. Garcia, "Cooperative Learning in Computer Programming: A Quasi-Experimental Evaluation of Jigsaw Teaching Strategy with Novice Programmers," *Education and Information Technologies*, vol. 26, no. 4, pp. 4839-4856, 2021, doi: 10.1007/s10639-021-10502-6.

[21] F. Demir, "The Effect of Different Usage of the Educational Programming Language in Programming Education on the Programming Anxiety and Achievement," *Education and Information Technologies*, vol. 27, no. 3, pp. 4171-4194, 2022, doi: 10.1007/s10639-021-10750-6.

[22] J. Leggett and G. Williams, "An Empirical Investigation of Voice as an Input Modality for Computer Programming," *International Journal of Man-Machine Studies*, vol. 21, no. 6, pp. 493-520, 1984, doi: 10.1016/S0020-7373(84)80057-7.

[23] S. Alharbi et al., "Automatic Speech Recognition: Systematic Literature Review," *IEEE Access*, vol. 9, pp. 131858-131876, 2021, doi: 10.1109/ACCESS.2021.3112535.

[24] R. M. Roy, B. Sabu, A. N, and P. A. R, "Voice Controlled Home Automation System," in *2021 7th International Conference on Bio Signals, Images, and Instrumentation (ICBSII)*, 2021, pp. 1-6, doi: 10.1109/ICBSII51839.2021.9445150.

[25] K. Baimirov, E. Mergengali, and B. Baimirov, "Overview of the Latest Research Related to Smart Speakers," in *2022 IEEE 7th International Energy Conference (ENERGYCON)*, 2022, pp. 1-5, doi: 10.1109/ENERGYCON53164.2022.9830196.

[26] P. Stavropoulou, D. Spiliotopoulos, and G. Kouroupetroglou, "Voice User Interfaces for Service Robots: Design Principles and Methodology," in *Universal Access in Human-Computer Interaction. Design Approaches and Supporting Technologies*, Cham, Denmark, 2020: Springer, pp. 489-505, doi: 10.1007/978-3-030-49282-3_35.

[27] O. Seeroo and G. Bekaroo, "Enhancing Student Support via the Application of a Voice User Interface System: Insights on User Experience," in *International Conference on Artificial Intelligence and its Applications (icARTi)*, Virtual Event, Mauritius, 2021: ACM, doi: 10.1145/3487923.3487936.

[28] P. T. Hakala, L. A. Saarni, R. L. Ketola, E. T. Rahkola, J. J. Salminen, and A. H. Rimpelä, "Computer-Associated Health Complaints and Sources of Ergonomic Instructions in Computer-Related Issues Among Finnish Adolescents: A Cross-Sectional Study," *BMC Public Health*, vol. 10, no. 11, pp. 1-8, 2010, doi: 10.1186/1471-2458-10-11.

[29] K. Keller, J. Corbett, and D. Nichols, "Repetitive Strain Injury in Computer Keyboard Users: Pathomechanics and Treatment Principles in Individual and Group Intervention," *Journal of Hand Therapy*, vol. 11, no. 1, pp. 9-26, 1998, doi: 10.1016/S0894-1130(98)80056-2.

[30] J. Ripat, E. Giesbrecht, A. Quanbury, and S. Kelso, "Effectiveness of an Ergonomic Keyboard for Typists with Work Related Upper Extremity Disorders: A Follow-up Study," *Work*, vol. 37, pp. 275-283, 2010, doi: 10.3233/WOR-2010-1079.

[31] D. Black, E. J. Rapos, and M. Stephan, "Voice-Driven Modeling: Software Modeling Using Automated Speech Recognition," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 252-258, doi: 10.1109/MODELS-C.2019.00040.

[32] A. Begel and S. L. Graham, "An Assessment of a Speech-Based Programming Environment," in *Visual Languages and Human-Centric Computing (VL/HCC'06)*, 2006, pp. 116-120, doi: 10.1109/VLHCC.2006.9.

[33] A. S. Elmaghraby, "Voice Recognition Applications for Programming Environments," in *IEEE Energy and Information Technologies in the Southeast*, 1989: IEEE, pp. 655-659, doi: 10.1109/SECON.1989.132471.

[34] M. Vacher et al., "Evaluation of a Context-Aware Voice Interface for Ambient Assisted Living: Qualitative User Study vs. Quantitative System Evaluation," *ACM Transactions on Accessible Computing*, vol. 7, no. 2, pp. 1-36, 2015, doi: 10.1145/2738047.

[35] O. Metatla, A. Oldfield, T. Ahmed, A. Vafeas, and S. Miglani, "Voice User Interfaces in Schools: Co-designing for Inclusion with Visually-Impaired and Sighted Pupils," in *2019 CHI Conference on Human Factors in Computing Systems*, Scotland, UK, 2019: ACM, doi: 10.1145/3290605.3300608.

[36] A. Wagner, R. Rudraraju, S. Datla, A. Banerjee, M. Sudame, and J. Gray, "Programming by Voice: A Hands-Free Approach for Motorically Challenged Children," in *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, Texas, USA, 2012: ACM, pp. 2087–2092, doi: 10.1145/2212776.2223757.

[37] D. D. L. Cordero, C. Ayala, and P. Ordóñez, "Kavita Project: Voice Programming for People with Motor Disabilities," in *23rd International ACM SIGACCESS Conference on Computers and Accessibility*, Virtual Event, USA, 2021: ACM, doi: 10.1145/3441852.3476516.

[38] O. Okafor, "Helping Students with Cerebral Palsy Program via Voice-Enabled Block-Based Programming," *ACM SIGACCESS Accessibility and Computing*, 2022, doi: 10.1145/3523265.3523267.

[39] G. Tisza and P. Markopoulos, "Understanding the Role of Fun in Learning to Code," *International Journal of Child-Computer Interaction*, vol. 28, pp. 1-10, 2021, doi: 10.1016/j.ijcci.2021.100270.

[40] L. Respondek, T. Seufert, R. Stupnisky, and U. E. Nett, "Perceived Academic Control and Academic Emotions Predict Undergraduate University Student Success: Examining Effects on Dropout Intention and Achievement," *Frontiers in Psychology*, vol. 8, no. 243, 2017, doi: 10.3389/fpsyg.2017.00243.

[41] M. B. Garcia, I. C. Juanatas, and R. A. Juanatas, "TikTok as a Knowledge Source for Programming Learners: a New Form of Nanolearning?," in *2022 10th International Conference on Information and Education Technology (ICIET)*, 2022, doi: 10.1109/ICIET55102.2022.9779004.

[42] S. Ruan, J. O. Wobbrock, K. Liou, A. Ng, and J. A. Landay, "Comparing Speech and Keyboard Text Entry for Short Messages in Two Languages on Touchscreen Phones," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1-23, 2018, doi: 10.1145/3161187.

[43] A. G. Hauptmann and A. I. Rudnicky, "A Comparison of Speech and Typed Input," in *Workshop on Speech and Natural Language*, Hidden Valley, Pennsylvania, 1990: Association for Computational Linguistics, pp. 219–224, doi: 10.3115/116580.116652.

[44] M. B. Garcia, " Hackathons as Extracurricular Activities: Unraveling the Motivational Orientation Behind Student Participation," *Computer Applications in Engineering Education*, pp. 1-16, 2022, doi: 10.1002/cae.22564