Vol. 4, No. 2 | Jul – Dec 2020

Analyzing the Effects of Error Messages Presentation on Debugging and Programming

Muhammad Shumail Naveed[1], Muhammad Sarim[2]

Abstract:

Programming is a fundamental skill of computer science students. However, it can be troublesome to learn. It is notable that programming error messages can be hard for beginners to comprehend, hampering progress and prompting disappointment. Effectively, translating compiler error messages is significant to rectify errors and advance toward victory in programming. However, these messages are often hard to understand and pose an obstruction to progress for many beginners. Descriptive messages are helpful for students in the beginning phase of learning a programming language. In this article, the effect of error message presentation on debugging and programming score is analyzed. The controlled experiment suggests that the presentation of error messages can have a strong effect on debugging score and also helpful to increase the programming score. However, no correlation between debugging score and programming score is identified. On the whole, a positive impact of the descriptive error messages is observed during the study.

Keywords: *compiler error messages; Java; introductory programming; debugging*

1. Introduction

Introductory programming has reliably been a center of computing science education [1], and one of the careers rewarding discipline [2]. The career openings of software development will increase through 2026 [3], however, learning to program is still a hard challenge for beginners and therefore high dropout is observed in the introductory programming courses [4].

Introductory programming courses include theory and practice of programming language with an integrated programming environment (IDE). Many programming environments are available and used in the educational context. The knowledge of IDEs is essentially important for students because there is an evidence of an association between the programming environment and learners metacognition [5].

Studies on the individual elements of programming environments remain an important topic, particularly from pedagogical aspect. One area of programming environment that has obtained much consideration in recent years is error messages. The errors are encountered during the process of developing code. When the error occurs, the developers expect an error message that describes what

[1]Department of Computer Science & Information Technology, University of Balochistan Quetta, Pakistan
[2]Department of Computer Science, Federal Urdu University of Arts, Science & Technology Karachi, Pakistan
Corresponding Author: mshumailn@gmail.com

has gone erroneous and attempt to correct the errors.

It is widely recognized that presentation of error messages had a significant impact on program comprehension. Errors may or may not cause compilation failure and the approaches for handling the two categories of errors fundamentally vary essentially. While runtime errors are critical, various studies center primarily on compile-time errors. Beginners may battle with the both types of errors; however, compile-time errors are more critical in that a program that compiler doesn't execute is unable to give the learners any important outcome of their efforts.

One of the numerous challenges' novices confronts from the begin are famously enigmatic compiler error messages, and there is an evidence on these challenges since 1965 [6]. It is widely observed that compiler error messages are frequently ambiguous, loose, confounding and erroneous, particularly for novice programmers. For beginners who are new to programming, learning the structure of a programming language can be troublesome, especially when the messages they get are confounding. Traver [7], identified that, the inadequately designed error messages affect the novices more negatively.

The analysis of errors made by beginners is of extensive enthusiasm of researchers in the domain of computing education; a comprehension of the errors that novices will in general experience, and how they manage them, is valuable in fitting teaching method and programming environments [8]. Since 1970's, it has become apparent that by and large, compiler error messages were not good as intended. Study of error messages in COBOL established that their feedback was not useful for users, especially beginners. As the education in computer science turned out to be increasingly broad, Pascal secured its position as the primary programming language for teaching. Brown examined the issues with the error messages in Pascal, seeing them as insufficient [9]. Similarly, a study on C language [10], explored the error messages, and gave vital knowledge into the developing notions over poor error messages. Denny et al. [11] identified that for over 20% of errors, the messages delivered by the Java compiler were not adequate to effectively distinguish the related error.

Beginners learn programming in different kinds of environments, from plain command-line to trade strength environments. These programming environments can contrast significantly in the presentation of error messages. Some environments provide very typical error messages, whereas some support very descriptive error messages.

Recently, some programming environments and compiler designers have shown an enthusiasm for giving descriptive error messages expected to be increasingly usable than previous. These studies are significant as connections can be drawn among compiler error messages and performance of users in programming. However, no notable study has mutually analyzed the impact of the error message presentation on debugging and programming. Likewise, no formal study has been found that examine the correlation between adeptness in debugging and programming proficiency.

The study presented in this article has three objectives. The first is to analyze the impact of error messages presentation on the capacity of debugging the programs. Specifically, the comparative analysis of enhanced compiler error messages and conventional compiler error messages in a controlled experiment. The second objective is to identify the effect of error messages presentation on the programming score. The third objective is to examine the association between adeptness in debugging and performance in programming. To the best of our knowledge no rigorous study with these objectives has been reported. This article lays out as follows. Related work is presented in section 2. Research method is described in section 3. Section 4 presents the results and discussion and followed by the conclusion.

2. Literature Review

The compiler error messages have been investigated in different dimensions and series of landmark studies on error messages have described blended findings. Dong and Khandwal [12], analyzed the effect of cosmetic changes on the ease of use of error messages. During study three visual dialects of the illustrative error message in a topical user interface framework are created. An online experiment which is based survey-based questionnaire included 52 participants. The results described that cosmetic changes to the introduction of an error message can have a strong effect on its usability.

Denny et al. [11] used CodeWrite a web-based environment wherein novices attempts a range of exercises that expect them to develop the body of a function in Java. The study of controlled experiment found no impact of extended compile time error message on the several metrics: number of submissions and the number of endeavors required to settle the most common syntax errors.

Becker examined the adequacy of enhanced compiler error messages [13]. This study focused on Java and employed an educational called Decaf, particularly developed for the research. The foremost thought that impacted the plan of Decaf was that Java compiled error messages might, and ought to be progressed upon. Decaf utilized the necessary information to develop more particular and accommodating enhanced compiler error messages which are displayed to the client. Two groups of around 100 subjects participated in the analysis. The study recorded 48,800 errors constructing a bunch of 74 different compiler error messages. The controlled group reported 32% less error than the control group and so demonstrates the viability of enhanced compiler error messages.

A landmark study of Traver [7] on the issues of compiler error messages described that the most recurrent errors do not fundamentally epitomize the complex errors to fix. Additionally, errors are not as it were due to the lack of information, or misguided judgments, but due to incidental slips. The study also identified that the significance of the quality of compiler error messages as suitable messages support the beginners to not essentially make arbitrary modifications to handle the error. In addition, useful error messages can decrease the workload of educates clarifying the same error messages again and again to understudies.

Nienaltowski et al. [14], analyzed the messages delivered by various compilers for different programming languages, and clustered them into three groups, and associated the degree of experience and error type with response and performance. The examination included two groups of subjects taking a basic programming course; they utilized messages in these three styles to troubleshoot incorrect code. The outcomes demonstrate that more definite messages don't really streamline the comprehension of errors yet that it makes a difference more where data is put and how it is organized. However, another study [15] on error messages reported that from their examination with C programmers that the individuals who extended feedback required less assistance from the tutors.

Becker et al. [16], analyzed the impact of enhanced error messages on the performance of novice programmer. The study adopted a different strategy by determining what number of syntax errors are fixed by novice programmers while investigating programs. During analysis, the impact of enhanced compiler error messages in a control experiment is analyzed where novices were given the task of correcting syntax of non-translated program code they didn't compose. A significant positive effect is found on the number of errors corrected, as well as the frequency of particular error types; however, no critical impact on the number of non-translating submission or novices scores.

Watson [17] presented BlueFix, a web-based environment amalgamated with BlueJ IDE and intended to help novices in diagnosing and repairing the errors. In contrast to other methodologies, BlueFix suggests a feedback approach dependent on systems joined from the human computer interaction and educational spaces, which can offer different novices with dynamic degrees of help dependent on their compiler behavior. An assessment was conducted recommending a 19% improvement and uncovering that students viewed the tool positively.

Karvelas et al. [18], investigated the programming behavior of novices in Java. During the study, Blackbox data is used and BlueJ versions 3 and 4 are analyzed. These two versions vary radically in terms of message presentation and behavior of compilation. The study identified that compilation method and presentation of error messages have a significant effect on the behavior of novice programmer.

On the whole the compiler error messages have been investigated in an endeavor to ease the issues and hardness that programming understudies proceed to confront. Several studies have been attempted to analyze the impacts of error messages with different approaches and expanding observation. To date the results of several studies are inconclusive whereas many studies have reported positive results. It is vital to note that for the foremost part, these researchers have been utilizing diverse metrics.

3. Design & Method

The investigation of programming errors made by beginners is of significant intrigued to researchers of computing education. A comprehension of the errors that novices will in general experience, and how they manage them, is valuable in fitting pedagogical method and educational programming environments. The study aims to analyze the effect of the compiler message presentation on program debugging, programming score and correlation between debugging abilities and programming performance.

The experiment was conducted on the undergraduate graduate (n = 44) of computer science in 2018. This study centers around Java, one of a most famous programming language for instructing beginners to the program and one of the most well-known language utilized in industry. It ought to be noticed that the decision of Java as an introductory programming language is not without criticisms, and other languages like Python has gained popularity as an introductory programming language.

A total of 44 subjects participated in the study and divided control and treatment group (each with 22). Subjects voluntarily participated in the study. The course was delivered by the same instructor, content and teaching strategy. The subjects were new to Java programming yet comprehended fundamental programming concepts.

Virtually, the stratified sampling is used to group the participants of the study. The participants were carefully clustered in two subgroups in a way that both groups share the same characteristics. In order to ensure the equivalence of sample groups the age, previous background of programming and basic mathematical skills of participants are considered while grouping the subjects. These characteristics are considered because they a notable impact on programming performance and may affect the result of the study. The participants of both groups of study were equivalent in terms of age as identified by independent sample t-test which shows no significant difference in the age of participants of control group (M=23.05, SD=2.89) and treatment group (M=22.73, SD=2.41) conditions; t(42) = 0.397, p=0.694. Similarly, the independent sample t-test on the pretest score of programming and mathematical skills shows no significant difference in the participants of control group (M=34.45, SD=17.52) and treatment group (30.36, SD=10.65) conditions; t(42) = 0.936, p = 0.355).

The course comprised of 14 lectures and included a weekly session of computer laboratory in which subjects attempted a series of programming problems and illustrated to the instructor. The study used BlueJ 2.0 to introduce programming to control group, whereas BlueJ 4.0 is used for the treatment group. The basic level of error messages in BlueJ 2.0 is the key reason for its selection for control treatment, whereas the descriptive error messages of BlueJ 4.0 is the pivotal motivation behind its selection for the treatment group.

There are multitude of programming environments for Java and some of them are BlueJ, Eclipse, NetBeans and IntelliJ IDEA. For presented study the BlueJ is selected because it is developed typically for beginners as identified by Becker et al. [19] and Yan [20]. Simplicity, visualization and interaction are the pedagogical features of BlueJ that makes it more superior over other environments. Unlike other environments, BlueJ support syntax highlighting and offers persistent pictorial feedback which allows beginners to visualize the execution path and the current state of the code in the program code [21]. Alkazemi and Grami [22], analyzed that BlueJ is more suitable for students than another environment.

During the study two initial sessions were devoted to the theory and introduced the fundamental concepts of programming to the participants. From third session the elementary programming with laboratory section was started and the data of students were properly logged for analysis. In every laboratory session a prewritten source code was provided to both groups of study and participants were asked to debug the provided programs by identifying and correcting the errors. The same practice was followed in every laboratory session and a debugging score of each participant is logged by enumerating the pre-existing errors corrected by novices while debugging programs. The detail of the debugging score of the control group is shown in Table I.

TABLE I. Debugging score of the control group

Students	**Session**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	Theory	Theory	10	8	21	15	12	18	41	48	60	15	47	61
2	Theory	Theory	21	29	31	30	33	41	35	57	65	62	53	70
3	Theory	Theory	11	16	20	10	16	19	20	37	34	60	19	55
4	Theory	Theory	4	6	15	20	5	12	17	19	12	22	30	24
5	Theory	Theory	4	19	19	25	32	28	21	12	28	33	42	63
6	Theory	Theory	16	21	10	12	6	16	19	32	19	18	32	23
7	Theory	Theory	9	18	25	26	18	37	23	43	59	50	37	45
8	Theory	Theory	8	11	17	19	16	36	30	43	31	52	27	59
9	Theory	Theory	16	11	15	27	5	32	27	32	42	17	41	62
10	Theory	Theory	10	6	12	11	12	21	28	20	30	40	29	33
11	Theory	Theory	11	15	17	21	29	21	23	48	35	44	31	27
12	Theory	Theory	15	9	30	21	14	11	24	52	57	29	22	15
13	Theory	Theory	5	6	14	16	17	20	19	33	41	34	18	29
14	Theory	Theory	8	18	17	14	22	20	35	33	60	59	27	28
15	Theory	Theory	4	14	6	14	15	13	37	16	61	56	30	34
16	Theory	Theory	15	19	23	28	31	11	16	33	24	25	32	45
17	Theory	Theory	6	14	26	23	32	24	31	38	27	15	40	50

18	Theory	Theory	4	8	7	19	20	14	21	25	18	28	26	33
19	Theory	Theory	13	21	30	10	8	22	26	22	22	17	23	53
20	Theory	Theory	19	19	19	28	27	23	32	47	30	36	30	60
21	Theory	Theory	4	13	23	26	27	34	25	44	61	42	28	25
22	Theory	Theory	15	18	24	28	37	42	45	53	60	62	51	66

A clear variation of debugging score is observed in the control group. A minimum score of a participant in the study is 186 whereas the maximum score is 527. The accumulated debugging score of participants in the control group is 7109. The debugging score of subjects in the treatment group is Table II.

TABLE II. Debugging score of the treatment group

Students	**Session**													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	Theory	Theory	10	8	21	15	12	18	41	48	60	15	47	61
2	Theory	Theory	21	29	31	30	33	41	35	57	65	62	53	70
3	Theory	Theory	11	16	20	10	16	19	20	37	34	60	19	55
4	Theory	Theory	4	6	15	20	5	12	17	19	12	22	30	24
5	Theory	Theory	4	19	19	25	32	28	21	12	28	33	42	63
6	Theory	Theory	16	21	10	12	6	16	19	32	19	18	32	23
7	Theory	Theory	9	18	25	26	18	37	23	43	59	50	37	45
8	Theory	Theory	8	11	17	19	16	36	30	43	31	52	27	59
9	Theory	Theory	16	11	15	27	5	32	27	32	42	17	41	62
10	Theory	Theory	10	6	12	11	12	21	28	20	30	40	29	33
11	Theory	Theory	11	15	17	21	29	21	23	48	35	44	31	27
12	Theory	Theory	15	9	30	21	14	11	24	52	57	29	22	15
13	Theory	Theory	5	6	14	16	17	20	19	33	41	34	18	29
14	Theory	Theory	8	18	17	14	22	20	35	33	60	59	27	28
15	Theory	Theory	4	14	6	14	15	13	37	16	61	56	30	34
16	Theory	Theory	15	19	23	28	31	11	16	33	24	25	32	45
17	Theory	Theory	6	14	26	23	32	24	31	38	27	15	40	50
18	Theory	Theory	4	8	7	19	20	14	21	25	18	28	26	33
19	Theory	Theory	13	21	30	10	8	22	26	22	22	17	23	53
20	Theory	Theory	19	19	19	28	27	23	32	47	30	36	30	60
21	Theory	Theory	4	13	23	26	27	34	25	44	61	42	28	25
22	Theory	Theory	15	18	24	28	37	42	45	53	60	62	51	66

Like control group, a variation in a debugging score of participants in the treatment group is clearly observed. The minimum score of participants in module is 298 and the maximum is 666. The accumulated debugging score of participants in the treatment group is 10012.

For comparative analysis the descriptive statistics of the debugging score of each group are calculated and results are shown in Table III.

The highest mean score is observed in a treatment group and similarly the highest debugging score in the study is also identified in a treatment group. For further analysis of debugging score the normality tests are conducted with SPSS. The Shapiro-Wilk test applied to the debugging score of control group confirms a normal distribution, W(22) = 0.92, p = 0.08. Similarly, a Shapiro-Wilk test on debugging score of treatment group identify a normal distribution, W(22) = 0.95, p = 0.36. The Kolmogorov-Smirnov test also identifies a normality in a debugging score of the control group (W(22) = 0.16, p = 0.15) as well in a treatment group (W(22) = 0.17, p = 0.11).

Table III. Descriptive statistics of debugging score

Group	Mean	Median	Std. Dev.	Min	Max	Range	Skewness	Kurtosis	Total
Control	323.14	324.00	80.573	186	527	341	0.911	1.586	7109
Treatment	455.09	447.50	92.838	298	666	368	0.589	0.380	10012

The distribution observed in the debugging score of both groups of study is graphically represented with detrended normal Q-Q plots shown in Fig. 1.

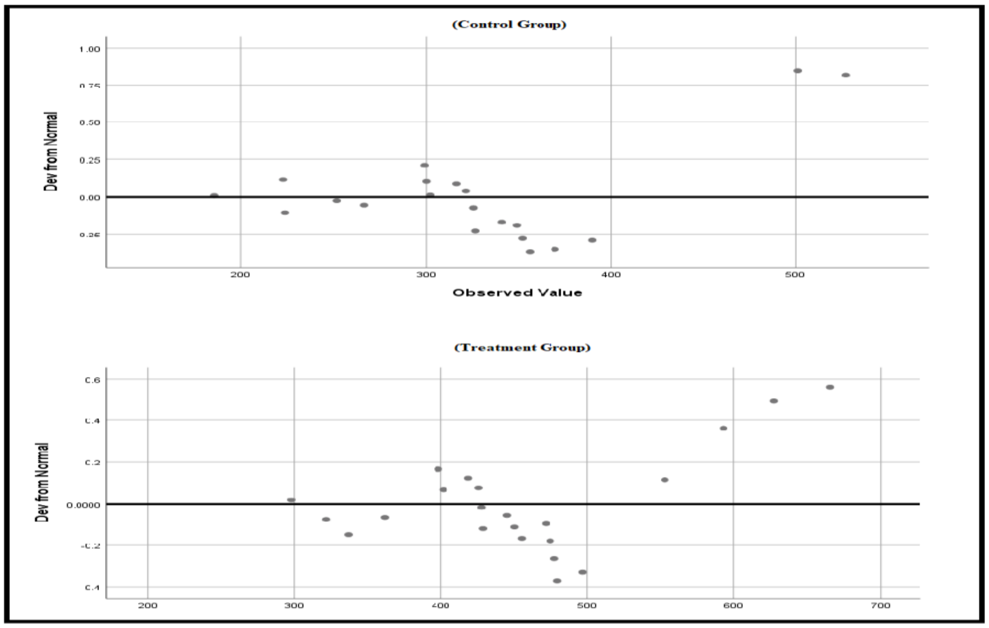

Fig 1. Detrended Normal Q-Q Plots of Debugging Score

The symmetric trend shown in the detrended normal Q-Q plots shows the normality of a debugging score of both groups in the study. The difference between score of two groups in study is further elaborated with boxplots shown in Fig. 2.

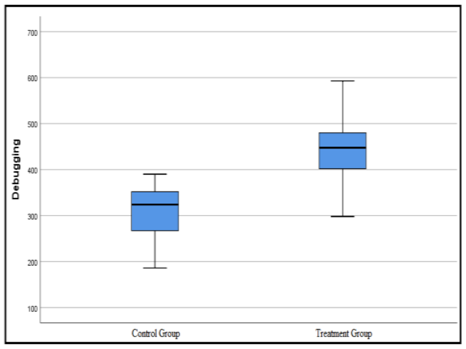

Fig. 2. Boxplots of Debugging Score

The quartiles of debugging score represented in the boxplots demonstrate the high performance of treatment group over the control group. The independent sample t-test was conducted to compare the debugging score of treatment group and the control group. There was a significant difference in the scores of a control group (M=323.14, SD=80.57) and treatment group (M=455.09, SD=92.84) conditions; t (42) = 5.03, $p < 0.05$.

During study the impact of error messages presentation on programming score is determined by conducting laboratory-based exam of contents covered during the module. During internal examination the participants of both groups are evaluated with same programming tasks and results are shown in Table IV.

Table IV. Detail of programming score

Group	Mean	Median	Min	Max
Control	49.50	54.00	15	75
Treatment	58.27	59.50	23	90

High mean score in programming is observed in a treatment group. For further analysis of programming score the normality tests are conducted. The Shapiro-Wilk test applied to programming score of control group confirms a normal distribution, W(22) = 0.95, p = 0.31. Similarly, a Shapiro-Wilk test on a programming score of treatment group identify a normal distribution, W(22) = 0.97, p = 0.74. The Kolmogorov-Smirnov test also identifies a normality in a programming score of the control group (W(22) = 0.13, p = 0.20) as well in a treatment group (W(22) = 0.15, p = 0.20).

The distribution observed in a programming score of both groups is graphically represented with detrended normal Q-Q plots shown in Fig. 3.

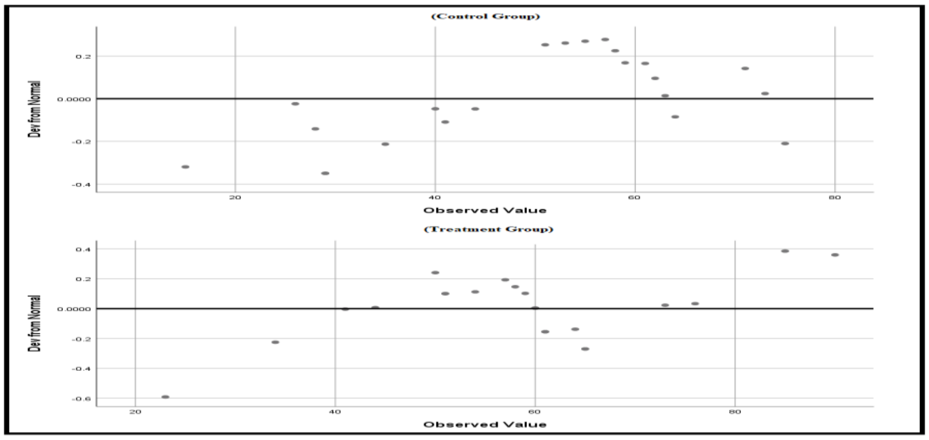

Fig. 3. Detrended Normal Q-Q Plots of Programming Score

The symmetrical trend revealed in the detrended normal Q-Q plots show the normality of programming score of both groups in the study. The difference between score of two groups in study is further elaborated with boxplots shown in Fig. 4.

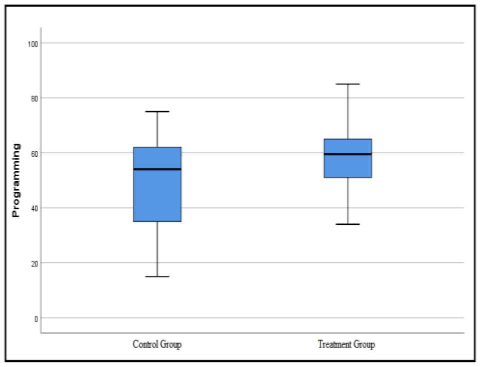

Fig 4. Boxplots of Programming Score

The quartiles represented in boxplots demonstrates the high score of programming in the treatment group than the programming score of a control group. The independent sample t-test was conducted to compare programming score of treatment group and a control group. There was no significant difference in the scores of a control group (M=49.60, SD=16.94) and treatment group (M=58.27, SD=15.32) conditions; t (42) = 1.80, p = 0.079.

A bivariate Pearson correlation was run to determine the relationship between debugging score and performance score. There was a moderate and positive correlation between debugging and programming, which was statistically significant (r = .453, n = 44, p = .002).

4. Discussion

The study presented in this article aims to analyze the effect of error messages presentation on the debugging score and programming score. The study also analyzed the correlation between the debugging score and programming score. A total of 44 subjects clustered in two groups. The BlueJ 2.0 is offered to the control group whereas, BlueJ 4.0 is offered to the treatment group. From the third session of a module the participants of both groups were given the problems of eliminating error from the source code they didn't develop and accumulated debugging score of 7109 is observed in control group and 10012 for the treatment group. The percentage difference of 33.91% between the accumulated scores of two groups suggests that presentation of error messages positively effects the adeptness in debugging. The t-test conducted on the debugging score of groups in a study described that descriptive error messages are helpful in debugging the program and certainly increased the adeptness in debugging.

The programming skill of participants in both groups is evaluated at the end of module by conducting a laboratory-based exam. The mean score of 49.50 is found in the control group, whereas 58.27 in a treatment group. The percentage difference of 16.27% in programming score of two groups suggests that presentation of error messages affects the programming abilities. However, t-test identified no significant difference between the programming score of a control group and treatment group which statistically described that the descriptiveness of error messages does not inevitably increase the programming score of novice programmers.

The correlation between debugging score and programming score is examined during the study. Technically, bivariate Pearson correlation identified a positive association; however, the relationship between variables is so weak, which signifies that adeptness in debugging does not reflect the proficiency in programming.

5. Conclusion

There are numerous challenges confronted by novices learning to program, and some are generally experienced as those in translating compiler error messages. These error messages are amazingly imperative as the novices' essential source of knowledge in their work, giving instantaneous feedback expecting to assist student find, analyze and rectify the errors. The presentation of errors has a profound effect on programming. In this article the effect of error messages presentation on debugging score and programming score is analyzed. The results suggest that descriptiveness of error messages significantly increased the proficiency of debugging the programs. Similarly, the programming score is increased by using programming environments that support descriptive error messages yet no statistical significance of descriptive error messages in improving the programming score is observed. Likewise, no strong correlation between the debugging score and programming score is found during the study. On the whole the study suggests the positive impact of descriptive error messages in introductory environments. However, there are several threats to the validity of results. First, the experiment is conducted on a small sample of participants, so the different results may be obtained on large and different kind of samples. Second, the effect of error messages presentation is examined on a single programming language. Third, the severity of errors is not examined during the study. As for further work, the study would be repeated on a large class of samples by examining the error messages of different programming languages. Similarly, other techniques and methods like principal component analysis shall be unified for further comprehensive analysis.

ACKNOWLEDGMENT

The authors would like to acknowledge all the students who voluntarily participated in the study. Special thanks to Muhammad Tahaam and Muhammad Aayaan for their valued comments and encouragement that made this research possible.

References

[1] M. S. Naveed, M. Sarim, and K. Ahsan, “Learners Programming Language a Helping System for Introductory Programming Courses,” Mehran University Research Journal of Engineering & Technology, vol. 35, no. 3, pp. 347-358, 2016.

[2] M. S. Naveed, M. Sarim, and A. Nadeem, “C in CS1: Snags and Viable Solution,” Mehran University Research Journal of Engineering & Technology, vol. 37, no. 1, pp. 1-14, 2018.

[3] https://insights.dice.com/2019/01/03/software-developer-jobs-increase-2026/ (Last Accessed: 12th September, 2020)

[4] L. E. Margulieux, B. B. Morrison, and A. Decker, “Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples,” International Journal of STEM Education, vol. 7, pp. 1-16, 2020.

[5] J. Prather, R. Pettit, B. A. Becker, P. Denny, D. Loksa, A. Peters, Z. Albrecht, and K. Masci, “First things first: Providing metacognitive scaffolding for interpreting problem prompts,” in proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp. 531-537, 2019.

[6] S. Rosen, R. A. Spurgeon, and J. K. Donnelly, “PUFFT - The Purdue University Fast FORTRAN Translator,” Communications of the ACM, vol. 8, no. 11, pp. 661-666, 1965.

[7] V. J. Traver, “On Compiler Error Messages: What They Say and What They Mean,” Advances in Human Computer Interaction, pp. 1–26, 2010.

[8] D. McCall, and M. Kölling, “A New Look at Novice Programmer Errors”, ACM Transactions on Computing Education, vol. 19, no. 4, pp. 1-30, 2019.

[9] P. J. Brown, “Error messages: The Neglected Area of the Man/Machine Interface”, Communications of the ACM, vol. 26, pp. 246–249, 1983.

[10] S. K. Kummerfeld, and J. Kay, “The Neglected Battle Fields of Syntax Errors,” in proceedings of the Fifth Australasian conference on Computing Education. Australian Computer Society, pp. 105–111, 2003.

[11] P. Denny, A. Luxton-Reilly, and D. Carpenter, "Enhancing syntax error messages appears ineffectual", in proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, pp. 273–278, 2014.

[12] T. Dong, and K. Khandwala, "The Impact of 'Cosmetic' Changes on the Usability of Error Messages," Extended abstracts of the chi conference on human factors in computing systems, pp. 1-6, 2019.

[13] B. A. Becker, G. G., R. Iwashima, C. McDonnell, K. Goslin, and C. Mooney, "Effective compiler error message enhancement for novice programming students", Computer Science Education, vol. 26, no. 2-3, pp. 148-175, 2016.

[14] M. Nienaltowski, M. Pedroni, and B. Meyer, "Compiler error messages: What can help novices," in proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, pp. 168–172, 2008.

[15] E. Odekirk-Hash, and J. L. Zachary, "Automated feedback on programs means students need less help from teachers," in proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education, pp. 55–59, 2001.

[16] B. A. Becker, K. Goslin, and G. Glanville, "The Effects of Enhanced Compiler Error Messages on a Syntax Error Debugging Test," in proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 640-645, 2018.

[17] C. Watson, F. W.B. Li, and J. L. Godwin, "BlueFix: Using Crowd-Sourced Feedback to Support Programming Students in Error Diagnosis and Repair," Lecture Notes in Computer Science, vol. 7558, pp. 228-239, 2012.

[18] I. Karvelas, A. Li, and B. A. Becker, "The Effects of Compilation Mechanisms and Error Message Presentation on Novice Programmer Behavior," in proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 759-765, 2020.

[19] B. A. Becker, P. Denny, R. Pettit, D. Bouchard, D. J. Bouvier, B. Harrington, A. Kamil, A. Karkare, C. McDonald, P. Osera, J. L. Pearce, and J. Prather, "Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research," in proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, pp. 177-210, 2019.

[20] L. Yan, "Teaching Object-oriented Programming with games," in proceeding of sixth international conference on Information Technology: New Generations, pp. 969-974, 2009.

[21] K. V. Haaster, and D. Hagan, "Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool," Issues in Informing Science & Information Technology, vol. 1, pp. 455-470, 2004.

[22] B. Y. Alkazemi, and G. M. Grami, "Utilizing BlueJ to Teach Polymorphism in an Advanced Object-Oriented Programming Course," Journal of Information Technology Education: Innovations in Practice, vol. 11, pp. 271-282, 2012.