ISSN: 2708-7123 | Volume-**01**, Issue Number-**04** | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

Programmer Assisted Tool Impact on Static Error Handling Capability of Novices in Imperative First Programming Languages

ShafaqueSaira Malik[1], Muhammad Shumail Naveed[1], Furqan-ul-Haq Siddiqui[1], Attiq Ahmed[1],Ehsan Ullah[1]
[1]Department of Computer Science and Information Technology, University of Baluchistan, Quetta, Pakistan.

ABSTRACT—Learning and understanding the syntax of a programming language is an extremely ordeal for novice programmers majoring in computer science. Introduction to programming is offered as a core subject. Novices have to use IDEs to write their programs. These IDEs has a valuable impression of novice error handling skills as static error messages are represented as intricate compiler waffles, terms and puzzling sentences. Docile, easy to understand, simple error messages are of prissy importance to evaluate novice programming aptitude. This research represents the outcomes of programmer assisted tool in natural language for explication of static errors in an imperative first programming language like C. Programmer Assistant tool (PAT) represents natural language description/solution of static errors underpinning Human Computer Interaction (HCI) approach, in the IDE and work as an offline static code analyzer. To assess effectiveness of this PAT novices was directed to write programs in different IDEs first and later using this PAT. Frequency of static errors, error, problem-solving time was analyzed and compared. The result of this study depicts that use of the programmer assisted tool has deep impact on novice performance, motivation and learning outcome. The quantitative mathematical analysis of our study revealed programming assistant tools has significantly influenced programming and static error handling skills of the students majoring CS.

Keywords—First programming Language (FPL), Natural Language Framework (NLF), Novices, Error Message (EM), Programmer Assistant Tool (PAT)

I. INTRODUCTION

Syntax is a splendid annoyance for apprentice programmers. Learning programming language syntax is very problematic for novices and syntax of a programming language is the only hindrance to their best performance. Error messages related to the syntax are inscrutable and apprentices flop to grasp them. The factors like syntax, error handling and tenacity of errors dramatically disrupt the performance of novices. The syntax and semantic of any programming language have significant upshot on the performance, enthusiasm of pupils and as a result, novice disbursed their most of the time struggling with the grammar of the programming language and could not develop skills like problem resolution (Hooshyar et al. 2015). Self-directed error handling approach to resolve and illustrate static errors enhance better understanding of static errors in first programming language like C, study conducted by (Malik, Naveed, and Siddiqui 2020) indicates that error messages in natural language improved static error correcting time which has substantial influence on performance, enthusiasm, mental load of apprentices. If the error message is convoluted and difficult then it will not be comprehensible by novices and may often lead to erroneous track and thus engender hindrance in novice programmers and significantly hampers their learning aptitude and recommended that intricate obscurities in error messages lead to new errors(Marceau and Fisler 2009). A case study was conducted on compiler errors by (Traver 2010) and concluded that error messages are puzzling and obstinate to comprehend by novices. Debugging unswervingly impact performance and inspiration of novices. The error messages do not essentially specify the appropriately cause of the error as a consequence novice attempt hard to retort to these error messages and they often prompt pupils to inapt edicts and cause even more errors often. It was noted by (Schliep 2015) that prominence of error cause uncertainty and corresponds to uncertain location and recommended that error messages should be novice amicable and embodied in conversant vocabulary rather than compiler waffles, intricate terms and inexact sentences. This PAT was developed to facilitate novices in programming. Natural language based error descriptions can assist novice programmers to better understand and remove errors in their programs before compilation has deep impact on learning outcomes of novices in a first programming language like C, this case study was conducted on students enrolled in CS

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

(Malik, Naveed, Siddiqui, et al. 2020). Most of the tools with enhanced error messages are developed online and mostly for java programming and not for imperative programming languages like C and most of the available static code analyzers are online like RepelIT, CShell, GDB, Clang. The prime input of this work is the analysis of paraphernalia on novices in imperatives first programming language like C by facilitating them with PAT, serving as an offline static code analyzer espousing natural language framework for description and resolution of static errors in natural language. Error messages in this framework are user sociable with meek vocabulary rather than compiler jargons, intricate terms and abstruse sentences. It is used as a coaching tool in the initial programming course to endorse surface and deep learning in novices through resolution of static errors former to compilation. The purposes of this PAT are to condense cognitive load to detect errors while writing code to solve programming problems thus promoting self-directed error resolution, self-efficacy, and self-learning of novices, abbreviate student preservation /abrasion in computer science, and at the same time will consequence in amplifying in interest, performance and programming skills of novice programmers.

II. LITERATURE REVIEW

Novices in their first introductory programming course encounter a mental effort and cognitive load. Learning programming languages is very difficult for most of the students and has considerable impact on enrollment and retention (Cooper, Dann, and Pausch 2003).Malik, Naveed, and Siddiqui (2020) conducted a case study on the students enrolled in CS and concluded that novice spent considerable amount of time to locate, fix static errors in their early programs.Novice programmers must deal with delusions, debugging, problem-solving and they often face coding misconceptions a due to lack of knowledge or simply incorrect assumption (Algaraibeh, Dousay, and Jeffery 2020). Novices are very frail in problem solving and analysis and it is very much exaggerated due to complex environments and syntax of the language and thus introductory programming language is a hurdle, the syntax and semantic of any programming language has significant effect on the performance and motivation of students and as a result novices spent their most of the combating with the grammar of the programming language and could not develop skills like problem solving (Hooshyar et al., 2013). Over the year's efforts were made to remove syntax and many tools are developed to achieve like Alice and Scratch. It is considered that the syntax of computer programming is "austere and stern" since it trails stanch rules that do not tolerate for maneuver and deviation. The semantics and syntax error are hectic for an inexperienced person. It has been observed that syntax error reported by the compiler at a location within the program is many lines away from the source of the error as a result novice faces high altitude of discomfort and cognitive effort and cognitive load. This is frustrating for the coder and, for the students, as a result, they may drop the program all together(Porter and Calder 2004). The most vital points of contact between the system and programmer in programming environment are the error messages. Students have problems reading, writing, tracking, designing, debugging simple code segments (Siti Rosminah and Ahmad Zamzuri 2012).Insufficient debugging skills aide nuisance and introduction of new error.Naveed, Sarim&Nadeem (2018) introduced the concept of learning mini language before learning introductory programming language with complex syntax and semantics and was called as LPL (Learners Programming Language) as a ZPL (Zeroth Programming Language). Inconsistency of errors and symptomatic messages generated by compiler is often hard nut to crack and is worsened when same error messages are generated for different errors and hence complier generate confusion and uncertainty accompanied by inconspicuousness to remove error and results in de-motivation, frustration and poor performance of novices. Encountering the same compiler error iteratively consecutively is tough majority of the students spend majority of their time on the understanding of error and hence they have to spend extra time on resolving syntax errors (Denny, Luxton-Reilly, and Carpenter 2014).Over the years many tools are developed to resolve the issue and to assist

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

debugging by integrating enhanced error messages. M.Koorsse et al (2015) conducted a survey and concluded that use of programming assistance tools (PAT) in environment for teaching programming may allow novices to be more confident in learning programming. Compiler error messages are snappy and for many novices, it is hard to write syntactically accurate programs (Ben-ari 2007). For novice programmer's scarce compiler error messages are challenging and are main assistance for debugging (Karvelas, Dillane, and Becker 2020). Juded (2005) reported that commercial compilers generate uninformative and sometimes miss leading error messages. Description of static errors has considerable impact on the novice performance (Malik, Naveed, and Siddiqui 2020).

III. METHODOLOGY

The data related to syntax error encountered by novices was collected in programming classes. Frequency of most common errors was calculated. Most frequent static syntax errors encountered by novices during their programming assignments along with their frequency are displayed in the table 4.1.

Table 4.1. Frequency of Errors in Programming Assignments

Syntax Errors	Number of Times Occurrence	Frequency
; missing	185	7.80%
undefined symbol	176	8.40%
} Missing	44	4.30%
) Expected	82	4.90%
Return missing	48	2.39%
Incorrect declaration of variable	86	4.89%
Method not defined	43	4.20%
(Expected	44	4.60%
<Identifier expected	43	4.49%
Uninitialized variable	30	2.34%
Undeclared variable	44	6.81%
Illegal start of expression	44	6.32%
Parameter type mismatch	45	4.03%
If (a<b);	46	2.20%

Evaluation

The evaluation of PAT was done by implementing it among novice groups "treatment group" along with "control group" using Turbo C IDE. Time was recorded from the error logs maintained in PAT and also manually novices were required to note time for writing programs in Turbo C, then both were compared.

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

Table 4.2 Most Frequent Static Syntax Errors and Simple Error Messages in PAT

The study was conducted on novices in imperatives first programming language like C by applying PAT and analyzing its impact on the performance, motivation, cognitive load, debugging time. Error messages in this framework are user friendly with simple vocabulary rather than compiler jargons, complex terms and ambiguous sentences. This framework has been used as a teaching tool in an introductory programming course. This paper represents investigations related to the effectiveness of this model, facilitating novice debugging, promoting self-directed static error resolution. The benchmark used to measure novice performance is error diagnostic time, number of errors, error resolution time. It is inferred that if error diagnostic time, increase performance will decrease and vice versa. Performance is inclusively related to the time required to write the programs and it is influenced by static error description. It was observed that increase in error resolution time decreased performance on the contrary fast correction of static errors, reduces over all static error diagnosis time subsequently increasing performance and motivation. We inferred that performance is mutually exclusively related to static error diagnostic time. We calculated frequency of static errors and their debugging time in 45 different programs assigned to novices in both PAT and Turbo C. We calculated error frequency and time to write the program. The errors occurred was maintained in error logs for each programming assignment, later groups were interchanged to evaluate their performance. Most common errors and enhanced error messages for them are listed in the table 4.2 that PAT addresses.

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

	Error	Description	Error message
Conceptual	;	Missing terminator at the end of the statement	The put semicolon '; ' in the end of statement to remove the error"
	>)}	Missing closing or opening braces	Put '} ' in the end of statement to remove the error" Put "{"to remove the error
	Printf Print Print	P of ptintf in uppercase Missing f from print statement	• Write printf in the place of print to remove the error • Write lowercase ' p ' in Printf to remove the error
	Undefined variable Undefined symbol Incorrect declaration of variable	Undefined variable	• The variable is not defined please define it or check the spelling
	Initialize int from char	Wrong initialization	• Assign Integer Value or Change int to char to remove the error"
	) Expected	Missing parenthesis	• Put ') ' at the end of statement to remove the error
	(Expected	Missing parenthesis	• Put "("at the beginning of statement to remove the error
	<Identifier expected	Identifier not defined/ missing	• Place identifier/ variable • The variable is missing in the statement either declare missing identifier /variable
	Uninitialized b=variable	The variable has not been initialized	• Please assign value to variable to remove the error
	Method not defined		• Correct the spelling Define header file at the beginning of your program #include<stdio.h or conio.h>
	Return missing	Return statement missing	• Make function void if not returning or passing any value For example, void main (void) • Type "return ()"; before closing}
	Illegal start of expression		Start expression by using"_" underscore Or by using any letter remove numbers, special characters and predefined symbols to remove the error
	Parameter type mismatch		Change the type and of parameter in the function to remove the error
	If (a<b);		Remove error by removing terminator semicolon ";" at the end of if statement

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

Framework For PAT

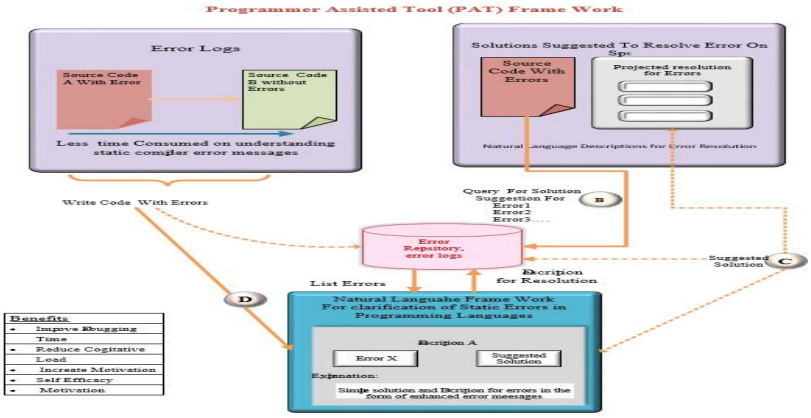
Performance of novices is dependent upon errors. Performance directly influence motivation to learn programming, motivation is dependent upon performance. Novice programming is influenced by the errors encountered and resolved, errors have an impact on the performance of novices and performance has impact on the motivation of novices. Performance of novices is dependent upon errors. Performance directly influence motivation to learn programming, motivation is dependent upon performance. Novice programming is influenced by the errors encountered and resolved, errors have an impact on the performance of novices and performance has impact on the motivation of novices. Fig 4.3 Illustrating conceptual framework for PAT augmented with enhanced error message in natural language along with the solution to correct and resolve the error which is in simple and easy to understand format for novices and a result they have to spend less time on understanding and resolving errors.

Fig 4.3 Conceptual Framework for PAT

The main aim of this study was to affirm impact of error message in simple natural language on novice programmers and to investigate response of novices to error messages in

imperative first programming language like C and how this has profound insinuation on their programming ability and final scores, along with performance, diagnosis and resolution of static errors, impact of error message on novice response and correction time to static errors, program writing time, understanding of syntax errors in a better way than conventional tools used to write programs and influence of enhanced error message framework in natural language and solution to static errors before compilation on perseverance, performance of novice students and learning has induced high notch of efficiency and confidence in programming ; ensuring low cognitive load and elevating interest in learning programing. The novices were given 45 programs ranging from simple to complex and were divided into two groups, they were required to write programs individually and later in groups. Novices were instructed to note down the number of errors they encountered in individual programs along with total time they required to write, debug, run their programs. The abettors in two groups were novices who attempted FPL like C, enrolled in computer science majors. Novices were divided into two groups control and treatment groups. They Control groups was instructed to use turbo C and the treatment group was instructed to use PAT. After few months' groups were interchanged. Live data were collected. Novices were required to use these tools in their class assignments and programming assignments.

In the control group, grouping was done randomly with keeping in view class performance of novices, the group included blends of weak, lethargic learners and good learners with good programming skills. Some of the partners were changed and reassigned to others and were allowed to work in collaboration on the basis of demographic factors like living in a dormitory / hostel with the same background,



ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

language and remote areas. The novices constantly were required to stay in the same group throughout the semester. Data was combined and compared from each group. Novices were assigned different programming assignments in-class which they were required to complete within class and out-class assignments were required to be submitted and presented within the period of one week. Attendance required in all groups was necessary.

Students in both groups were required to submit 15 home assignments and 10 to 15 in-class practice programming projects, both types of assignments were given scores for functionality, rate of error and their correction and error handling, readability and time estimated time to write, execute and debug particular code, debugging time for static errors was also calculate for each programming task. Novices were required to submit their error logs in different programming assignments in both the. Static syntax error understanding, error solving knowledge has deep impact on performance, cognitive load and motivation of novices. It was observed that novices working in control groups took more time to write, resolve static errors and rate of static typing errors was high, number of self-assumed errors was also high furthermore same error occurrence was also high and more oftenly they fail to understand what actual error was as a result they were lethargic, bored, fed-up with less self-confidence, high cognitive stress in the solutions they have developed, most of the time they were observed besieged with error rectification and on average spent more time. It was observed that control group members were often carped that they can't grasp what was taught in class during their projects on the contrary treatment group, however performed better than due to proper highlighting of errors, simple error messages and correction suggestion. Both groups were given 15 programs to write, afterwards that 6 programs were given to each group with errors in order to analyze how much time they take to debug programs, as expected control group debugging time was longer than the treatment group. Novices submitted and completed their assignments in both groups. Novices attempted their terminal exams independently. Data was collected regarding their scores in FPL, programming time, debugging time.

.

IV. DATA ANALYSIS& RESULTS

Theresults of this study show that while at first, the novices struggle with syntax which is is typical with learning a new language. Course outline of FPL was same in all the groups. Supposition of this study was set keeping in sight hypothetical research perspective of novices to succeed. Terminal scores of all the groups were collected, the time consumed to write program in class and at home was analyzed, debugging time in both groups was compared and contrasted to assess our supposition that accomplishment and performance of novices in FPL is influenced by static errors, error diagnostic time. The cognitive load of novices is considerably affected by error messages generated whenever a static syntax error encountered. The table 5.1. Depicts total number of novices in

both groups, female participants, the number of dropouts, table 5.2. Depicts average scores in both groups

Novices Average Scores in FPL	
Control Group	**PAT Group**
56.81632653	79.55556

Table 5.2 Average Score in FPL

Novices Enrolled in FPL	
Control Group	**PAT Group**
Total number of novices = 59	Total number of novices =54
Total number of dropouts=8	The total number of dropouts=14
Total number of males=51	Total number of males=48
Total number of females=8	Total number of females=6

Table 5.1 Novices enrolled in FPL

Performance in terms of Debugging Time

It was suggested by (Donna Teague and Paul Roe 2008) that learning programming is affected by lack of self-assurance, concentration. We collected data on programming activity from. Novices in both the groups were given 45 programs from easy to complex. The average time required to debug a

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

single program was calculated and compared in both the groups and it is depicted in table 5.3

Novice, Average Debugging Time in FPL	
Control Group	**Treatment Group**
13.31295 minutes	12.91556 minutes

Table 5.3 Average Debugging Time

Re-composition of compiler error messages in natural language has a strong association with the performance of novices. It is very astonishing to infer from results that students in the treatment group performed better as compared to male /female counterparts in the control group. The time required for each program is on the average greater in control group even for simple programs like a pyramid of stars as compared to the treatment group. Novices from aforementioned groups performed better in solo in class for each of given assignments with a smaller number of errors and time then novices in the control group, however when they were asked to switch to PAT their performance enhanced both in grades and in debugging time, their overall debugging time improved in PAT due easy error messages and solution suggestions. They strive less hard to correct errors. The average time required to debug each program was less in the treatment group furthermore overall average debugging time, the average time required to write, compile, debug programs and total number of the overall error count was also less in the PAT as compared to control group, results are illustrated in following table 5.4

	Control Group	**Treatment Group**
Average Debugging Time to Fix Static Errors	13.58 Minutes	10.46 Minutes
Average to Write, Compile, and Debug Programs	14.7 Minutes	12.8 Minutes
The total number of static errors Fixed	3875	2819

Table 5.4 Average Debugging Time and Total Count of errors in 10 Lectures

The benchmark used to measure novice performance is error diagnostic time, number of errors, error resolution time as depicted in table 5.4. It is inferred that if error diagnostic time, increase performance will decrease and vice versa. Performance is inclusively related to the time required to write the programs and it is influenced by description of static error. It was observed that increase in error resolution time decreased performance on the contrary fast correction of static errors, reduces over all static error diagnosis time subsequently increasing performance and motivation. We inferred that performance is mutually exclusively related to static error diagnostic time and outcomes of the study are also reflected in the Fig 5.1, 5.2, and 5.3

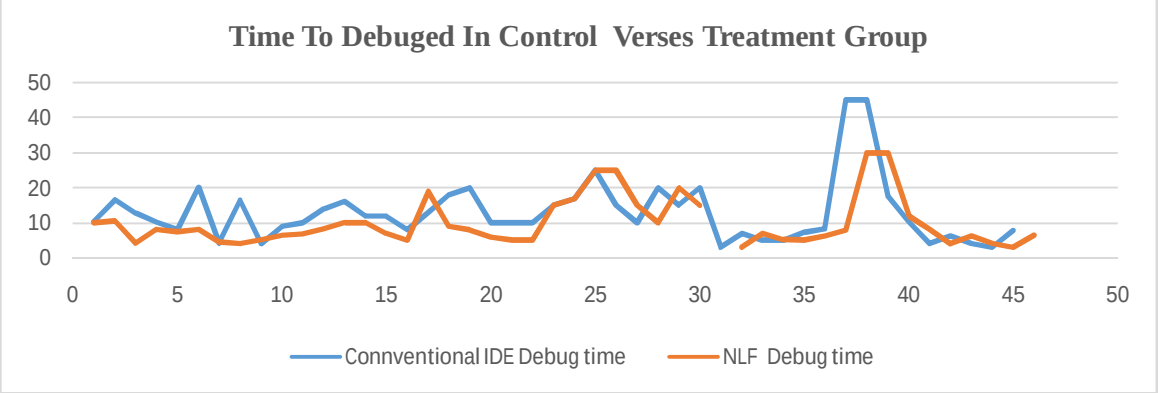
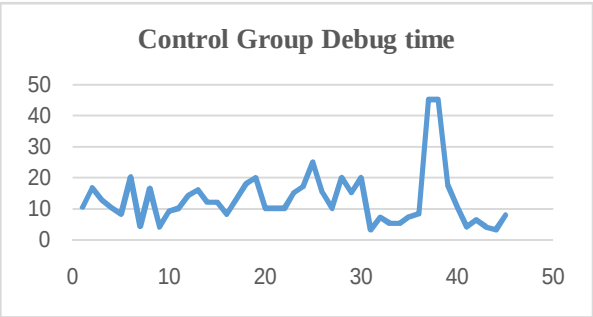


Fig 5.1 Time To Debug Programs In Control Group Using Conventional IDE Like Turbo C And Control Group Using NLF Based PAT

ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150



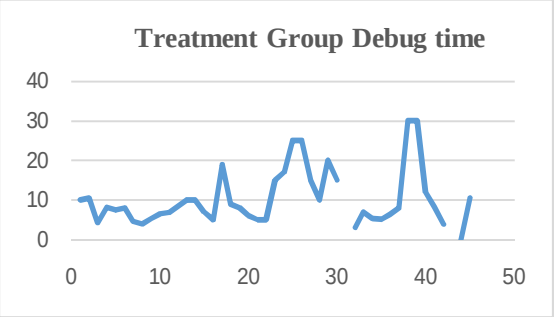


Fig 5.2 Control Group Debug TimeFig 5.3 Treatment Group Debug Time

The fig 5.1 and fig 5.3 clearly illustrates that novices took less time to debug in the NLF PAT as compared to conventional IDE with complex error messages. By the completion of FPL, novices in treatment group performed better with elevated interest. Quality of programs produced by them was much better, with fewer errors and more readable. The fig 5.4 and 5.5 represent frequency of errors novices encountered during the semester in 10 lectures in FPL.

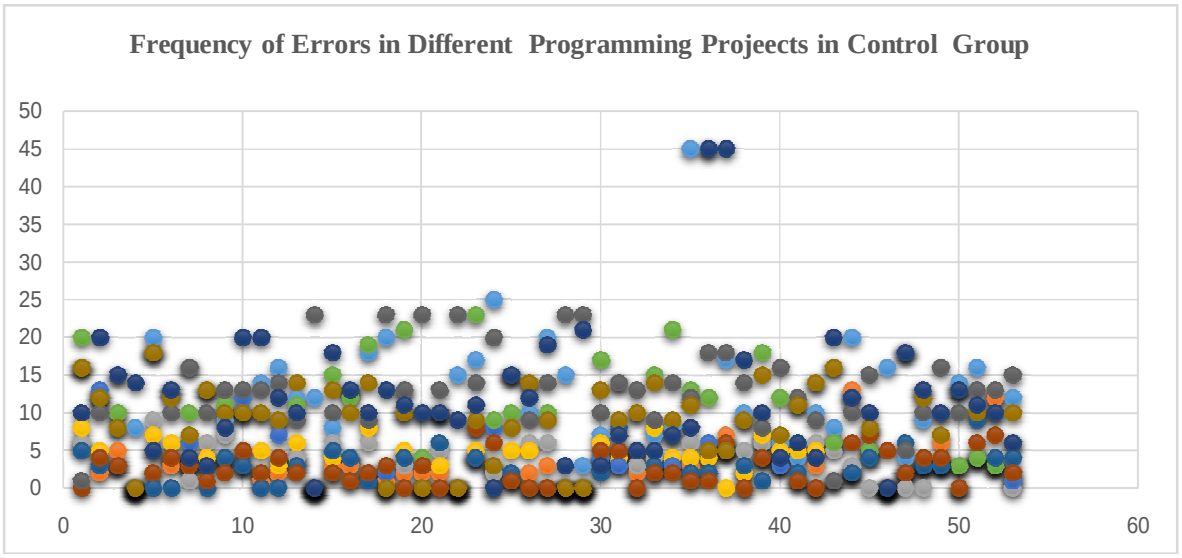
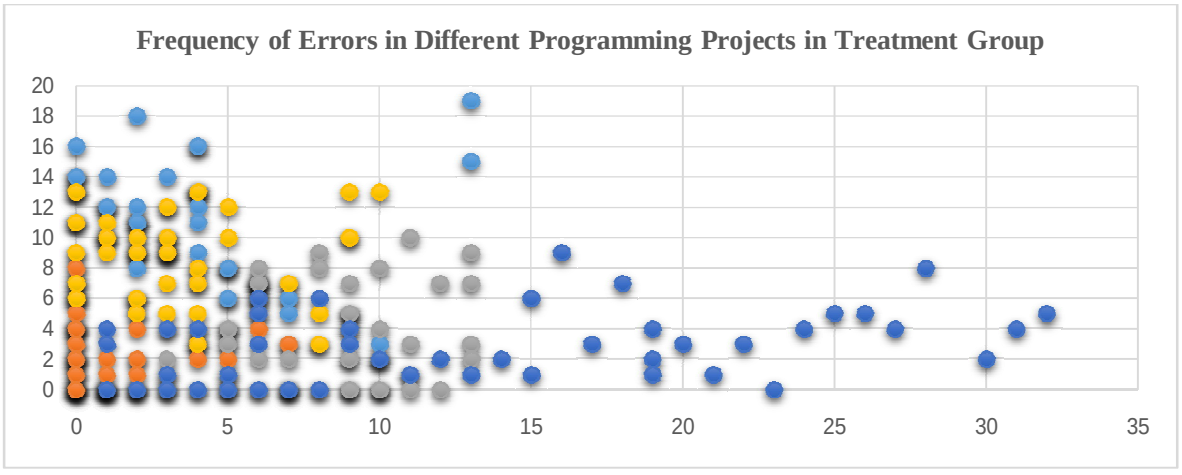


Fig 5.4 Frequency of Errors in Control Group



ISSN: 2708-7123 | Volume-01, Issue Number-04 | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

Fig 5.5 Frequency of errors in treatment group

Static syntax errors	Control Group average weightage of error	Treatment Group Average Weightage of Error
; missing	11.63870968	12.45122384
undefined symbol	10.60645161	7.484923732
}missing	9.677419355	10.64207166
) expected	8.980645161	9.08123448
return missing	7.535483871	6.385242994
incorrect declaration of variable	8.103225806	5.285562256
method not defined	7.535483871	4.398722951
(expected	5.987096774	4.753458673
<identifier expected	3.561290323	4.895352962
uninitialized variable	7.587096774	9.223128769
undeclared variable	4.516129032	6.207875133
illegal start of expression	4.903225806	6.775452288
parameter type mismatch	4.95483871	6.881873005
if(a<b);	4.387096774	5.853139411

Table 5.5 Weightage Frequency of Errors

Frequency of error occurrence is less in different programming projects less in treatment group using PAT as compared to control group as depicted in the fig 5.3 and 5.4. We conducted pre-test to check the performance of novice in programming skills along with their age. T-test conducted on age shows that there was no significance in the age of both the groups. The **t-value** is -0.7 and the p-value is .22. The result is not significant at $p < .05$**.** The group sampling for equivalence was done on the basis of academics and age. Both the samples were divided into two groups. The academic performance level was analyzed on the basis of programming algorithmic skills and mathematical skills in the pre-test. The T-test conducted on the performance of subjects shows that there was a significant difference on the score of the control group and treatment group such that **t-value = 1.67** and **$p < .05$**. It is concluded that over all PAT is useful, handy for the majority of novices. Through this research it is certain that PAT augmented with natural language-based enhanced error message for handling static syntax errors in programming is especially advantageous for novice in CS1 as it comprehensively addressed many considerable surfaces which prevents participation and progress of novices in computer science and programming. It is inferred that debugging becomes easy if error messages are in simple, easy to understand natural language then compiler jargons. Learning to program with determination is promoted hence ensuring inventiveness, firmness and effective software development in computer science majors and encourages novices to trail their potential programming careers in CS.

V. CONCLUSION

The results of this study specify that PAT used as a programming tool increase learning and improve performance and inquisitiveness of novices, and that this is of prim importance in order to enhance performance and inspiration of novices reliably on fixing static errors. We assume that PAT will benefit students to develop potential skills to overcome barriers in their first programming language course. It is inferred from results that understanding of error messages to resolve static errors plays a very imperative part in an engaging, indorsing, nourishing curiosity, enthusiasm of novices whether male or female in

learning. Programming and hence safeguarding real progress of programming and auspicious future in computer science.

The results of this study provide real indication that syntax errors handling in the programs effects performance of novices in their first programming course effectually and later on it will play significant role in future programming-oriented subjects and has profound influence on novice performance and is one of most effective pedagogical tools

that has deep impact and effect on the learning capability and performance of novices in CS1. Novices who were using PAT showed high competence in programming with a high notch of error resolution, good problem solving and produces better quality programs then others.

PAT is an effective tool in learning programming for novices and increase their skill in programming, although and it has profound outcome on the performance of novices in FPL.PAT has significant impact on error resolution, diagnosis of errors, understanding of errors, identifying classes of errors in early programming with ease.

Limitation of study

The limitation of this research that it is confined to only introductory /first programming course. This study was confined to imperative languages and not object-oriented languages. Demographic factors in the evaluation were not considered in this research and it was confined to static syntax errors.

ACKNOWLEDGMENT

The authors are obliged to the Department of Computer science for the benefaction and facility. The authors would also like to thank all the pupils who contributed to the study. We are thankful to Mr.Yashran and Mrs.MumtazIdress for their encouragement and support. We are thankful to Mr. Saeed Kaker, Mr.Umer for their support.

REFERENCES

Algaraibeh, San'a M., Tonia A. Dousay, and Clinton L. Jeffery. 2020. "Integrated Learning Development Environment for Learning and Teaching C/C++ Language to Novice Programmers." *Proceedings - Frontiers in Education Conference, FIE*.

Ben-ari, Mordechai Moti. 2007. "Compile and Runtime Errors in Java." 1–28.

Cooper, Stephen, Wanda Dann, and Randy Pausch. 2003. "Teaching Objects-First in Introductory Computer Science." In *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*.

Denny, Paul, Andrew Luxton-Reilly, and Dave Carpenter. 2014. "Enhancing Syntax Error Messages Appears Ineffectual." *ITICSE 2014 - Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference* 273–78. doi: 10.1145/2591708.2591748.

Donna Teague, and Paul Roe. 2008. "Collaborative Learning - towards a Solution for Novice Programmers." *Conferences in Research and Practice in Information Technology Series* 78:147–53.

Hartmann, Björn, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. 2010. "What Would Other Programmers Do? Suggesting Solutions to Error Messages." *Conference on Human Factors in Computing Systems - Proceedings* 2:1019–28. doi: 10.1145/1753326.1753478.

Hooshyar, D., R. B. Ahmad, M. Yousefi, F. D. Yusop, and S. J. Horng. 2015. "A Flowchart-Based Intelligent Tutoring System for Improving Problem-Solving Skills of Novice Programmers." *Journal of Computer Assisted Learning*, *31* (4): 345–61. doi: 10.1111/jcal.12099.

Karvelas, Ioannis, Joe Dillane, and Brett A. Becker. 2020. "Compile Much? A Closer Look at the Programming Behavior of Novices in Different Compilation and Error Message Presentation Contexts." *ACM International Conference Proceeding Series* 59–65. doi: 10.1145/3416465.3416471.

Malik, Shafaque Saira, Shumail Naveed, and Furqan-ul-haq Siddiqui. 2020. "Creation of CFG Based Natural Language Framework for Explication of Syntax Errors

in First Programming Language Featuring Novices." *LC International Journal of STEM* 1 (2). doi: 10.47150/jstem014.

Malik, Shafaque Saira, Shumail Naveed, Furqan-ul-haq Siddiqui, and Mohammed Umer. 2020. "Implementation of CFG Based Natural Language Framework in Description of Syntax Errors in Imperative First Programming Languages: A Case Study from University of Baluchistan." 8–17.

Malik, Sohail Iqbal, and Jo Coldwell-Neilson. 2017. "A Model for Teaching an Introductory Programming Course Using ADRI." *Education and Information Technologies* 22 (3): 1089–1120. doi: 10.1007/s10639-016-9474-0.

Marceau, Guillaume, and Kathi Fisler. 2009. "Mind Your Language: On Novices ' Interactions with Error Messages." 3–17.

Marceau, Guillaume, Kathi Fisler, and Shriram Krishnamurthi. 2011. "Measuring the Effectiveness of Error Messages Designed for Novice Programmers." *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* 499–504. doi: 10.1145/1953163.1953308.

Munson, Jonathan P., And Elizabeth A. Schilling. 2016. "Analyzing Novice Programmers' Response To Compiler Error Messages." *Journal of Computing Sciences in Colleges* 31 (3): 53–61.

Porter, Ron, and Paul Calder. 2004. "Patterns in Learning to Program: An Experiment?" *Proceedings of The Sixth Conference On Australasian Computing Education* (30): 241–46.

Robins, Anthony V. 2019. "Novice Programmers and Introductory Programming." In *The Cambridge Handbook of Computing Education Research*.

Sampson, Demetrios G., J. Michael Spector, Dirk Ifenthaler, Pedro Isaías, and International Association for Development of the Information Society (IADIS). 2017. "Proceedings of the International Association for Development of the Information Society (IADIS) International Conference on Cognition and Exploratory Learning in Digital Age (14th, Vilamoura, Algarve, Portugal, October 18-20, 2017)." *International Association for Development of the Information Society*.

Schliep, Paul A. 2015. "Usability of Error Messages for Introductory Students." *Morris Undergraduate Journal* 2 (2).

Siti Rosminah, MD Derus, and Mohamad Ali Ahmad Zamzuri. 2012. "Difficulties in Learning Programming: Views of Students." *1st International Conference on Current Issues in Education (ICCIE2012)* 74–78.

Traver, V. Javier. 2010. "On Compiler Error Messages: What They Say and What They Mean." *Advances in Human-Computer Interaction* 2010. doi: 10.1155/2010/602570.

BIOGRAPHY

Indicate the level of contribution of each author. All authors should include biographies with photo at the end of regular papers.

Personal profile which contains details about their email id, education, publications, research work, membership, and achievements with photo and will be maximum 100-150 words.

Shafaque Saira Malik was born in Quetta, Pakistan.She received the BCS. degree in computer science from the Allama Iqbal Open University, ISB, Pakistan and MCS from UOB, Quetta, Pakistan. ETE From AIT, Thailand and STKLC from Ehwa University, Seoul, South Korea.

In 2007, she joined the Department of Computer Science and Information Technology, University of Baluchistan as Lecturer. Her current research interests include programming, agile programming, pair programming, computer science education. She is Coordinator for University of Baluchistan, and is a Fellow of National Academy of Young Scientists Pakistan; She has received Technology award in Invention for Innovation Summit (2019).

ISSN: 2708-7123 | Volume-**01**, Issue Number-**04** | December-2020
LC INTERNATIONAL JOURNAL OF STEM
Web:www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

Furqan-ul-haqSiddiui was born in Mastung, Pakistan. He did graduation in Commerce from UOB, Quetta, Pakistan and MBA from UOB, Quetta, Pakistan. He did MS from Iqra University, Karachi, Pakistan,In 2007, joined Institute of Management Science, University of Baluchistan as Lecturer. He is now serving as Assistant Professor.

Atiq Ahmed completed his MS and PhD from the University of Technology of Troyes, France in 2007 and 2010, respectively. Currently, he is working as an associate professor at the Department of Computer Science and IT in University of Balochistan (Pakistan). He has also served as the Director of the office of Research, Innovation and Commercialization (ORIC) in the same institution from 2012-14. He has published his research works in several well renowned conferences and journals like IEEE LCN, AICT, IEEE Communication Surveys & Tutorials, Annals of Telecommunication, etc. He has participated in several projects funded by AgenceNationale de la Recherche, European Union, British Council, ICT R&D and HEC. He has served as a reviewer for various journals and conferences like Computer Networks, IEEE Globecom, ICC, VTC, Journal of information Technology, IEEE AICT, IEEE AICSAA, IEEE MASS, SRJ, AMRJ... His research interests include Internet of Things, service continuity in wireless networks, autonomic networks, SDN and 5G networks, wireless sensor networks, network intelligence with the multi-agent systems, intrusion detection, quality of service, TMN and cloud computing.

IhsanUllah is working as Assistant Professor in department of Computer Science & IT, university of Balcohistan, Quetta, Pakistan. He completed his Ph.D degree under the supervision of Dr. Guillaume Doyen and Dr. Dominique Gaiti from Universite de Technologie de Troyes in 2011.

He got his Master of Computer Science degree from the same university in 2008. His research interests include P2P networks, video streaming, user behavior, Quality of Service and intelligence in P2P streaming networks. He has published several international journal articles as well as has presented in prestigious conferences such as IM, AIMS and CNSM. He has served as a reviewer for several conferences and journals and has also evaluated national research projects proposals for funding approval submitted to ICT R&D Fund (IGNITE).