ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

CREATION OF CFG BASED NATURAL LANGUAGE FRAMEWORK FOR EXPLICATION OF SYNTAX ERRORS IN FIRST PROGRAMMING LANGUAGE FEATURING NOVICES

Shafaque Saira Malik[1], Shumail Naveed[2] & Furqan-ul-haq Siddiqui[3]
Department of Computer Science and Information Technology,
University of Baluchistan, Quetta, Pakistan
shafaque.malik@gmail.com, mshumailn@gmail.com

ABSTRACT—The intention of this research is to investigate effectiveness and impact of NLF for error messages on the performance, motivation, cognitive load of novices in FPL like C. This study analyzed the effectiveness of enhanced error messages in natural language on debugging .it is used as a teaching tool in introductory programming language. This research focus on use of natural language framework to illustrate errors, suggest proper solution thus ensures that usability of error messages effectively to facilitate debugging. This paper reports that self-directed static error resolution and illustration using natural language, enhanced understanding of static errors and decreased debugging time. CFG based NLF ensemble natural language description underpinning HCI approach in IDE for resolution of errors. We inferred that novices using NLF performed better in programming with good understanding of static error handling, error resolution ,NLF has valuable impression on novice learning outcomes The results of study indicate error messages in natural language augmented static error debugging time which has considerable impact on performance, motivation, cognitive load of novices.

Keywords—FPL, NLF, novices, CFG, PAT, performance, enhanced error message, motivation, cognitive load.

I. INTRODUCTION

Learning programming is royal pain in neck for novice programmers. The factors like syntax, error handling and resolution of errors significantly affect the performance of novices

II. LITERATURE REVIEW

It is easier to debug errors if novices have clear understanding of processing steps and outcome of problem statement (Iqbal & Coldwell, 2017).

(Ovsyannikov, M. K, Kasimov, D. R.,2014; Kuchuganov 2017) concluded that multifaceted development environment and difficult syntax of the programming language elevate poor problem analysis and solving abilities for novice as a result of which novices are less motivated, badly overstrained and fight tough battle to learn syntax of the programming language.(Denny et al., 2014) noted that learning syntax of any programming language is very difficult and introduced CodeWork incorporating concept of enhance error message to improve errors messages.

The performance, motivation of novice programmers is significantly affected due to difficult syntax/semantic of programming language as a result novice spent their most of the combating with the grammar of the programming language and could not develop skills like problem solving (Hooshyar, Alrashdan & Mikhak, 2015; Hooshyar, Alrashdan & Mikhak, 2017).

Error messages are of prim importance and serve as a tool for programmers to find and rectify their mistakes in the programs they coded, and if the errors messages are not helpful then learning programming becomes hard nut to crack, these messages are basic source to understand what is wrong in the program. (Schliep, 2015). Error messages should not contribute to confusion (Isa et al., 1983). (Marceau et al., 2013) suggested that complex compiler error messages are difficult to comprehend by novices and may often lead to wrong path and thus generate frustration in novice programmers and greatly hinders their learning ability. (Marchue et al, 2011) suggested that complex ambiguities in error messages lead to new errors. Mismatch feedback from compilers upon static error occurrence often is problematic for novices (Munson & Schilling, 2016) and these error messages are explanation where translation broke down, it very difficult to learn.

If syntactical order of the parser is violated then syntax errors occurs. Error massages related to syntax are enigmatic and novices fail to comprehend them, user friendly error messages provide aid to novices to learn programming language (Schliep, 2015). Correcting the syntax is the first step to learn debugging. Compiler error message usability is of prim importance to diagnose errors, without correcting syntax errors the process will not proceed and correcting error is crucial component of debugging process and therefore it cannot be overlooked (Kummerfield & Kay, 2003).

According to (Marceau et al, 2011) the error messages received by novices do not directly indicate original error and they also noted that there are several issues related to effectiveness of error messages for example error message does not reflect properly the actual error student have received.

(Traver, 2010) conducted case study on compiler errors and concluded that error messages are cryptic and cumbersome to comprehend by novices. Debugging directly influence performance and motivation of novices. The error messages do not actually indicate properly cause of error as a result novice strive hard to respond to these errors messages and prompt students to inappropriate edicts and cause even more errors often.

(Schliep, 2015) noted that highlighting of error is ambiguous; they should be user friendly and in simple vocabulary rather than compiler jargons, complex terms and ambiguous sentences.

This research focus on use of natural language to illustrate errors and suggest proper solution thus ensuring effectiveness of error message usability to facilitate debugging and thus ensuring elevated performance of novices. This research will promote self-directed error resolution and illustration using natural language; it will also help students to get more exposure of programming structures and debugging skills, it can be incorporated in language modeling for error resolution. It will ensemble natural language description underpinning human computer interaction (HCI) approach, in IDE for resolution of errors and is based on augmented context free grammar (CFG).

This research posits that natural language paradigm is good feature and can be ensemble in programming. A syntax / error is core problem for novices and is inherited in programming, each grammatical rule /syntax can be expressed if not followed properly in a program in natural language while typing source code and can be implemented as single unit in Integrated Development Environment (IDE).

Novices in their first introductory programming course encounters a lot of mental effort and there is lot of cognitive load, there is it is very excess cognitive load and ineffective learning. Pedagogical factors also play important role in choice of programming language by instruct. Learning programming languages is very difficult for novices and has "considerable effect on enrollment and retention for the programs" (Dann et al, 2006). (Hooshyar et al, Alrashdan & Mikhak, 2015; Hooshyar, Alrashdan & Mikhak, 2017) indicated that 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.

Syntax is extravagant twinge in the neck for neophyte programmers. Problem solving is very difficult and it is accompanied by new-fangled mode of thinking at the same time. Many tools are developed to remove syntax for example Alice and Scratch.

It is considered that the syntax for computer programming is "austere and stern" since it trails stanch rules that do not tolerate for maneuver and deviation. The semantic and syntax error are hectic for an inexperienced person. Debugging, resolving errors is frustrating for the coder and, for the students, as a result they may drop the program all together (Porter & Calder, 2004).

"Many efforts were depleted in order to make programming easier in introductory programming courses" (Anewalt, 2008; Daly, 2011). Students have problems reading, writing, tracking, designing, debugging simple code segments (Derus et al., 2012). Many modern IDEs provide support for learning the language and also syntax through code completion however there is little support for error resolution and correction. Debugging is intricate skill for novices. Complier messages are often scarce and ineduqateand and

syntax error is cause of disenchantment and barricade to students triumph (Denny et al, 2014).

(Naved, et al., 2018) introduced the concept of learning mini language called as LPL(Learners Programming Language) as a ZPL (Zeroth Programming Language).Learning mini language is advantageous for novices before learning introductory programming language with complex syntax and semantics, it will help novices to understand syntax of introductory programming language as it generates high level program equivalently from the source code in plain natural language and express syntax in the form of algorithmic way based on the computational statements.

Disparity of errors and indicative messages generated by compiler is often hard nut to crack and is exacerbated when same error messages are generated for different errors and hence complier generate perplexity and ambiguity accompanied by obscurity to eradicate error and results in de-motivation, frustration and poor performance of novices. Recognition and identification of errors cannot be automated. Repeated errors play role of best indicators for evaluating students' progress (Jedud, 2006). Many students are unable to relate the mistakes highlighted by complier to the mistakes they have actually have made (Mathew, 1984) .

This research will explore erudition difficulties students' encounter when studying introductory programming course and discovering features to develop a natural language error illustration and resolution tool by incorporating enhance compiler error messaging technique for novice programmers. The approaches like "syntax free" "problem solving" and "computing" were introduced in 1999. ClockIt was developed by in 2009, and Dr. Racket is also environment to assist novice programmers. Ratina was developed in 2009 and it focuses on the errors during compilation and execution. Codework was developed in 2018, it is "simplified development environment", it is web based and provide interface to execute and review student code. (Kyfonidis et al., 2017) developed block-based visual shell for C.

This research intent to delineate an appropriate tool which will serve as platform for on spot syntax error correction and will suggest on spot resolution of error novice programmers. If implemented, it will have significantly affected self-efficacy, motivation and self-learning of novices, condense student retention /abrasion in computer science, and at the same time will consequence in amplifying in interest, performance and programming skills of novice programmers. The intent of this research is augment compiler error messages with the aid of on spot natural language illustration of error and simple suggestion for resolution of errors. Syntax is a magnificent ache in the neck for neophyte programmers. Leaning syntax is very difficult for novices. For years a lot of research has been conducted and many efforts were made to develop tools that eliminate syntax, most famous Alice and Scratch .Novices in their first introductory programming course encounters a lot of mental effort and there is lot of cognitive load, there is excess cognitive load and ineffective learning .Novices experience difficulties when learning basic programming concepts in introductory programming languages reported (Xinogalos et al ., 2017).Writing code and following the syntax of modern day popular programming languages is intricate and convoluted. (Marceau et al., 2011) conducted study and their finding demonstrated that error messages drastically fail to convey information accurately to novices.

"Learning to programming languages is very difficult for novices and it is usually source of anxiety and trouble for many students enrolled in computer science and has considerable effect on enlistment and retention for the programs" (Dann, Cooper, & Pausch, 2006). The researchers conducted over the period of time indicate that learning first programming course is intricate, complex for many students (Ali & Shubra, 2010; Daly, 2011 & Kaplan, 2010).

The semantic and syntax error are hectic for an inexperienced person. Furthermore, a syntax error reported by the compiler may be at a location within the program that may be many lines away from the source of the error .Modern integrated environments facilitate novices by providing support for learning language and syntax through code completion and feeble support for problem solving and novices feel frustrated and fail to progress. Compiler messages are less helpful when represented to students and same error message is generated for different errors, diagnostic errors are ambiguous and generate confusion and are difficult to resolve as a result novice are frustrated and demotivated. It is not easy

to automate identification of errors. When execution and compilation fail diagnostic errors are generated and elusiveness exists between errors and diagnostic messages and most errors generated are syntactical (McCall & Kolling, 2014).

Introductory programming is complex and intricate, and many efforts over the years were depleted in order to make it simple to learn and to make introductory programming course easy for novices (Anewalt, 2008; Daly, 2011). Many new syntax-free programming languages are introduced, which are easy to learn for example Alice, Blackly, Bayou, Scratch, and Tinkle.

Due to difficult syntax, semantic of the language students are demotivated to learn programming as a result there is high failure rate, retention and drop out in computer science .Understanding grammar of any language is extremely hard for novices, studies show that students have to brawl syntax understanding combat with high cognitive effort and load. "Between the system and programmer in any programming environment, error messages are one of the most imperative points to contact and students have problems reading, writing, tracking, designing, and debugging simple code segments" (Rosminah, MD Derus & Ali, 2012). Many modern IDEs provide support for learning the language and syntax through code completion, however there is little support for error resolution and correction. Student written code is often filled with errors, and meager debugging skills escort nuisance and introduction of new error noted by (Murphy et al., 2008). Error messages are generated by compiler to help novives to loacte and correct errors,how ever complier messages are often scarce and ineduqateand and syntax error is cause of disenchantment and barricade to students triumph(Denny et al., 2014).

(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. Incongruity of errors and diagnostic messages is often hard nut to crack and hence complier error messages generate perplexity and ambiguity accompanied by obscurity to eradicate error and results in de-motivation, frustration and poor performance of novices. "Students expend greater part of their time on resolving syntax error" (Denny et al., 2014).Repeated errors play role of best indicators for evaluating students' progress (Jedud, 2006).Many students are unable to relate the mistakes highlighted by complier to the mistakes they have actually have made (Mathew, 1984),good feedback in the form of error messages are of prim importance for novice programmers and are also very crucial for them if they want to learn programming .Over the years many tools are developed to resolve the issue and to assist debugging by incorporating "enhanced error messages" like Bluefix was developed in 2012, an online tool integrated in BlueJ, HelpmeOut developed in 2010 that assist debugging of error messages by signifying resolution that peers have applied in the past however partial assessment was carried out. In 2003 pre-compiler tool called Expresso was introduced to scan java programs for 20 frequent errors and prided users with explanatory messages for errors. It was interactive tool and provided suggestions to correct the code however its assessment was left to future work. BACCI is a tool to assist programming through flowchart description for novices; Raptor is windows based application to enhance problem solving skills and avoid syntax errors.(Juded, 2006) perceived that " commercial compilers engender uninformative and sometimes misleading error messages and means of learning how to deal with them is more effectively needed". These messages are snappish and for many novices it is thorny to write syntactically accurate programs (Ben-Ari, 2015). Complex error messages are difficult to track. Explicit programming knowledge and programming proficiency is required by novice programmers in order to write programs effectively (Koorsse et al., 2015). For novice programmer's inadequate compiler error messages are challenging and are primary assistance for debugging (Becker, 2016). (Juded, 2005) reported that commercial compilers generate uninformative and sometimes miss leading error messages.

(Mclver, 2000) reported that novices are frustrated by unproductive errors and are responsible for not providing learning opportunities and syntax errors hamper learning as students are agitated.

Context-free grammars are associated with linguistics where they are used to illustrate the structure of sentences, phrases and words in a language. In computer science they are used recursively for defining programming language concepts and syntax and describe structure of the programming languages. A context-free grammar is based upon a simple, mathematically precise mechanism for describing how phrases in language are built and are simple enough to represent construction of any parser mechanism.

(Rohrmeier et al., 2016) suggested that Context free grammar (CFG) play very important role for describing syntax of any programming language and central feature associated with the words organization, contents of phrases are based on CFG. The grammar was adapted from (Brian W.& Dennis M. Ritchie, 1988.) for NLF for elucidation of static errors in programming language.

III. METHODOLOGY

The main aim of this study was to affirm impact of error message in natural language frame work(NLF) based on CFG for static errors in programming on novice performance and motivation and to investigate response of novices to error messages in 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., glimpse optimization of diagnostic syntax error time , and influence of enhanced error message frame work in natural language and solution to static errors before compilation on perseverance, perseverance and 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 participants were from CS1 who were enrolled in course of "introduction to programming" and C language as FPL belonged to three consecutive batches of undergraduates. The total number of participants was 700 divided into two groups (Conventional IDEs and NLF group) one group was instructed to use conventional programming tools and IDES like classical Turbo C, Code Block, Dev , other group was initially instructed to use aforementioned tools and then were switched to tools like GDB, RepelIt, C shell, CPP Check, and eclipse and later on was instructed to first program in conventional tool then use visual C and NLF based on CFG. In each CS1 batch strength of students was 70 except for the later 3rd batch were strength was reduced to 50 student's enrollments due to change in admission policy. In the each group total number of participants was 70 to 78 out of which 5 to 7 were female novices and rest male and total number of 25 females in total and 5 to 7 females novices in each group and rest were males, The abettors in two groups were novices who attempted FPL, enrolled in computer science and information technology majors, at the university of Baluchistan during 2016-2017,2017-2018,2018-19 academic year spanning over two major groups CS and IT. After few months' groups were interchanged and performance was analyses in terms of error handling, writing, compiling, debugging time of programs ranging from very easy to difficult programs in C language which helped us to analyze cognitive load, confidence performance and motivation of novices. Live data was collected.

Aforementioned tools were used to write programs by the novices enrolled in CS1 and they were required to use these tools in their class assignments and programming assignments where as other group required to work independently alone on their assignments and in-class practice programs in NLF based on CFG.

In the conventional IDEs group consortium was done arbitrarily with keeping in view class performance of novices, the group included blend 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 demographical factors like living in dormitory/hostel with same background, 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.

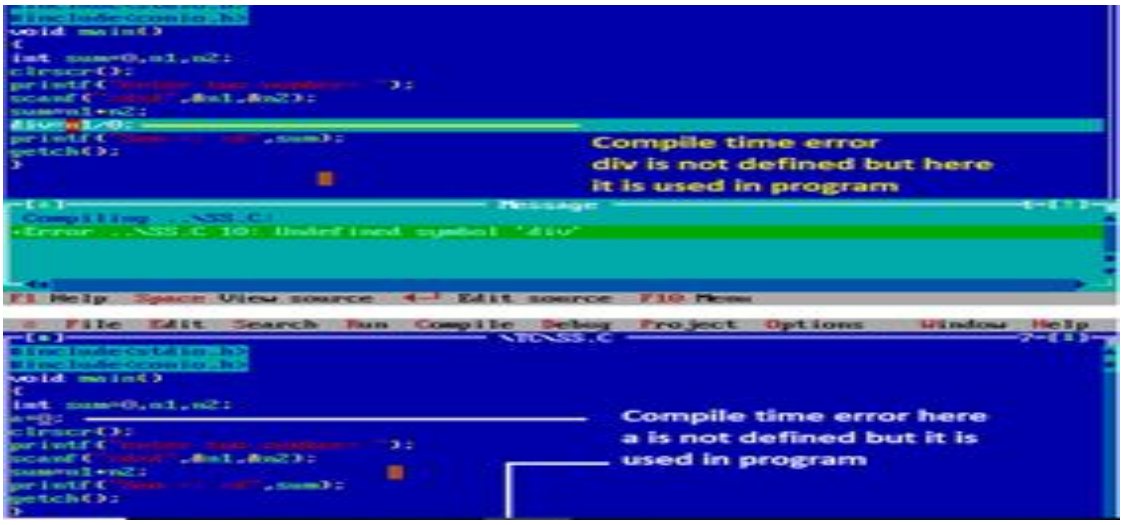
Novices were assigned different programming assignments in-class which they were required to complete

ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

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 also 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 all the groups along with the time scale. Clasp of programming, static error understanding, error solving knowledge unaccompanied and in groups has deep impact on performance, cognitive load and, motivation of novices. Students working alone and in conventional groups were demotivated, took more time to write, resolve static errors, rate of static typing errors was high, number of self-assumed errors was also high furthermore same error accordance was also high and most of the cases 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 has been observed that conventional IDEs group members were often carped that they can't grasp what was taught in class during their projects on the contrary other using online tool like GDB etc. was little relaxed but they often had issues with downloading tool ,understanding error messages, stressful structure, auto completion of code, it was noticed that when they were asked to use paper-pen approach to write programs they missed those structures from that were auto completed like auto accruing of {} in Code Block. The NLF group however performed better than the other group in both our CFG based NLF and Visual C, due proper highlighting of errors and enhanced error messages and correction suggestion. All 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 debag programs, as expected conventional IDEs group debugging time was longer than NLF group, however FPL group the error diagnostic time was least.

Novices despite of the fact submitted and completed their assignments in FPL in both groups, each one of them endeavored their terminal exams independently. Terminal and



practical results assess static error handling skills, programming knowledge, error handling, error understanding, debugging and capability to response errors in the code and programs. Data was collected regarding their scores in FPL, programming time, debugging time, survey was conducted to analyze their interest in programing, which were later compared and contrast for each of the groups. The error messages in classical IDE and NLF are depicted in the following figures.

Fig. 2.1 Example of Complex Error Message from Classical C IDE

The fig.2.1 shows complex error messages in classical Turbo C which are difficult to comprehend and are stated after compilation for syntax errors e.g. "statement missing " error is encountered when ever novice miss terminator at end of the

ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

statements that is ";" and usually highlight next line which is wrong highlight and it is often ambiguous for novices to fix error as a result they fix self-assumed error on next line which in fact generate another error on compilation , the error messages are complex as a result novice spend considerable amount of time to figure out and fix error as a result cognitive load increase and thus performance and motivation decrease. Error messages is the only way the novices respond to as feedback (Munson & Schilling, 2016), which are merely translation of code in a program by syntax analyzer, no token is generated by compiler but just a error message if it violates syntax and after checking syntax tree error is generated , it is usually in complex format as illustrate in figure 2.1 .The PAT we developed using CFG frame work generate errors messages in natural language and on spot for static error, CFG for this framework is adopted from was adapted from (W. Kernighan & Dennis M. Ritchie, 1988).

natural language framework for elucidation of static errors in programming language

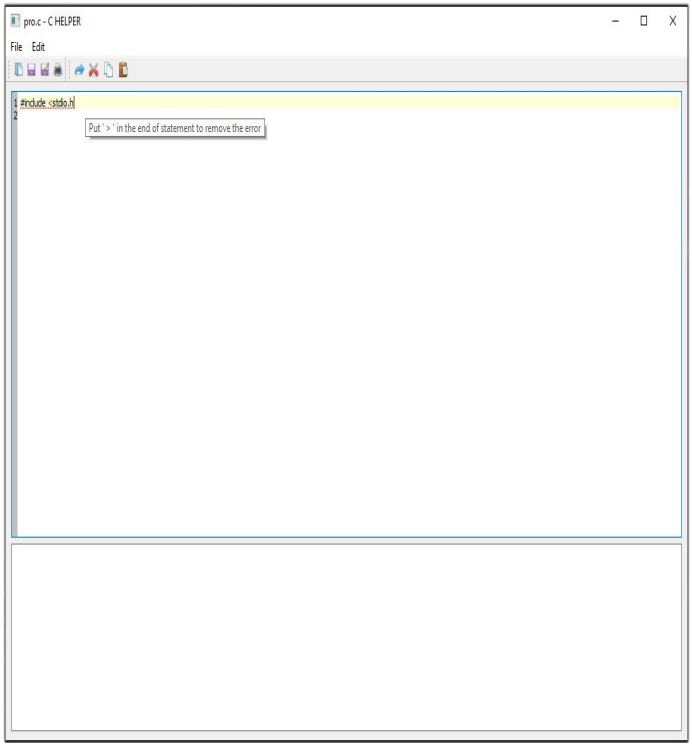
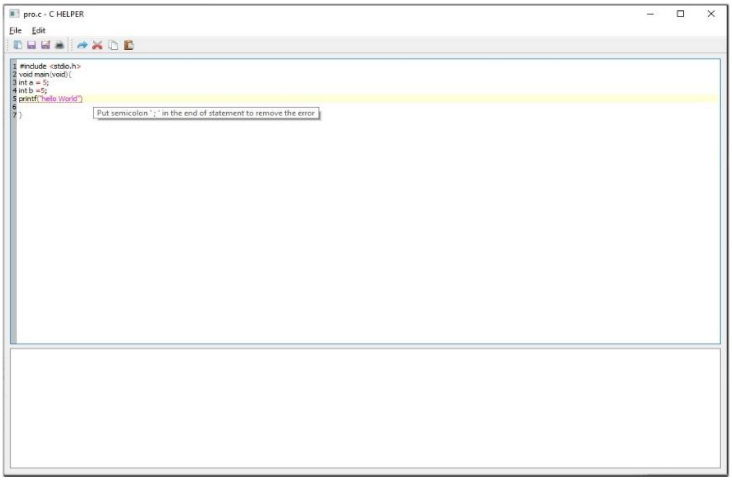


illustration of static error in NLF based on CFG



illustrating enhanced error message in natural language along with the solution

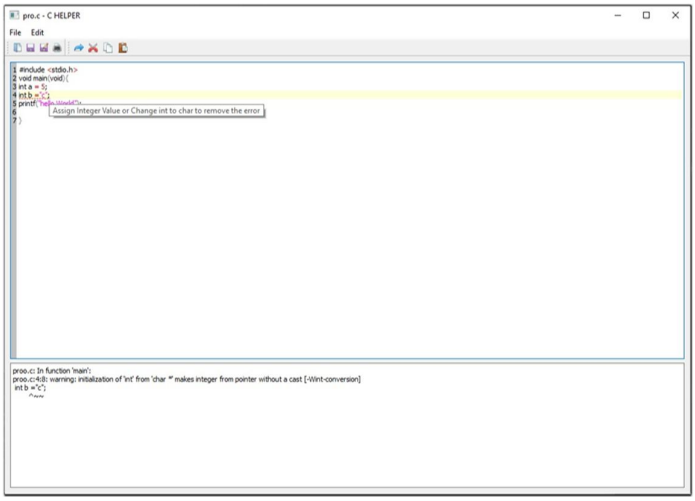


illustration of error and solution in nlf based on cfg

Illustrating 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 spent least time on understanding and resolving errors

IV. DATA ANALYSIS & RESULTS

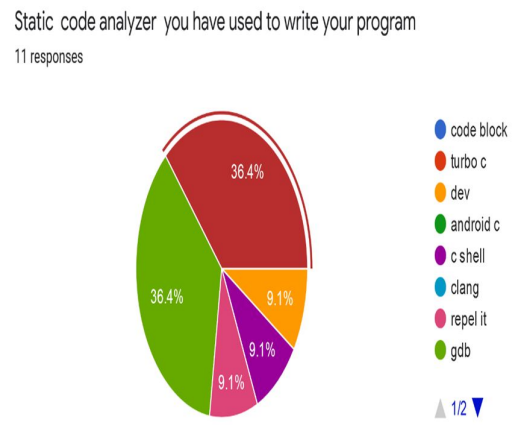
Course outline of FPL was same in all the groups. Hypothesis of this study was set keeping in sight hypothetical research perspective of novices to succeed. Terminal scores for all the groups were collected, time consumed to write program in class and at home was also analyzed for each of 15 programs, debugging time in different tools and NLF based

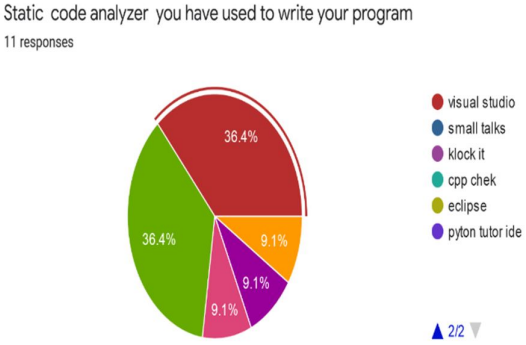
ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

on CFG were compared and contrasted to assess our hypothesis that moot accomplishment and performance and motivation of novices in FPL is influenced by static errors, error diagnostic time , and cognitive load of novices is considerably effected by error messages generated when even static syntax error encountered.

environments used

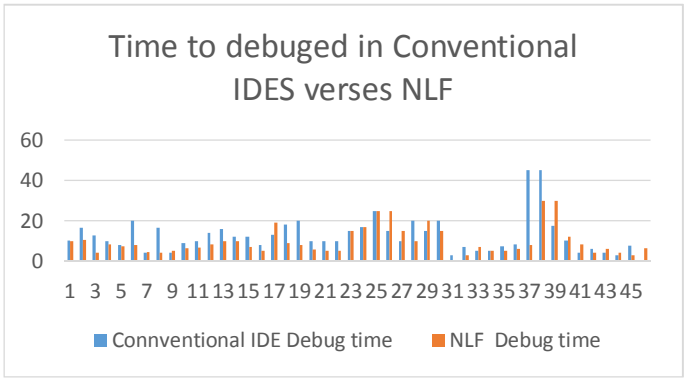
The following environments were used for programming groups of novices in FPL and then they were interchanged and were required to switch to NLF based on CFG and number of error fixes on the basis of enhanced error messages in natural language were analyzed and compared about 80% of the novices were of the view error messages were in natural language and easy to understand as compare to other environments. The IDES/ static code analyzers used by novices for writing programs are illustrated in figures





Performance in Conventional IDES verses NLF for elucidation of errors/debugging time

It was suggested by (Teague, D., & Roe, 2008,). that learning programming is affected by lack of self-assurance, concentration. We collected data on programming activity from 760 novices who were enrolled in CS1 and IT in 3 different batches in FPL.The novices used instructional programming environment called " NLF for elucidation of static errors in programming language" deigned based on CFG is deigned to vindicate static errors and represent error in natural language along with their solutions. Novices in both the groups were given 45 programs from easy to complex, the debugging time required to fix the errors was better in NLF based on CFG for static errors in programming language, the average time required to debug single program was calculated and compared with the debugging time in Conventional IDEs ,results are illustrated in the following figures

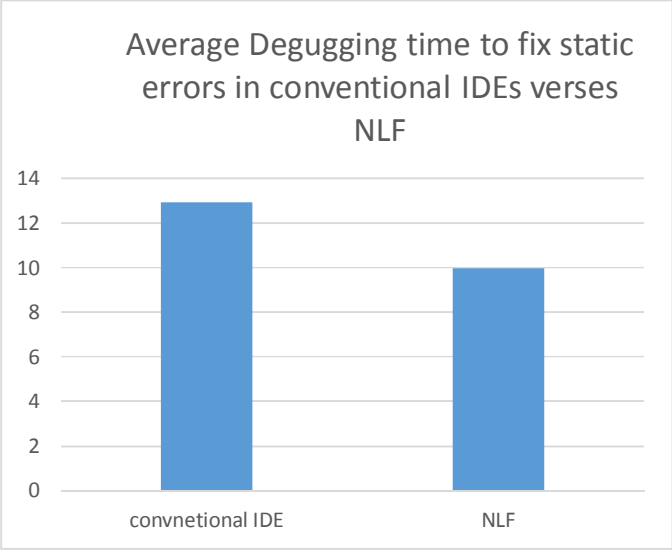


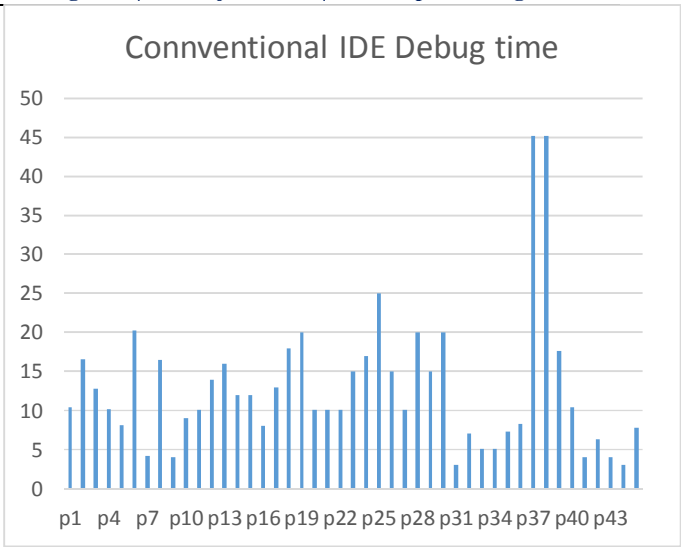
It is inferred from data collected that hypothesis H1 holds the performance, motivation and retention of novices are correlated with the induction of natural language in error resolution. Re-composition of compiler error messages in natural language has strong association with the performance of novices. It is very astonishing to infer from results that students in group NLF performed better as compare to male /female counterparts in CS1 in convention IDE group. The time required for each program is on average greater in Conventional IDE group even for simple programs like pyramid of stars as compared to NLF. Novices from aforementioned groups performed better in solo in class for each of given assignments with less number of errors and time

ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

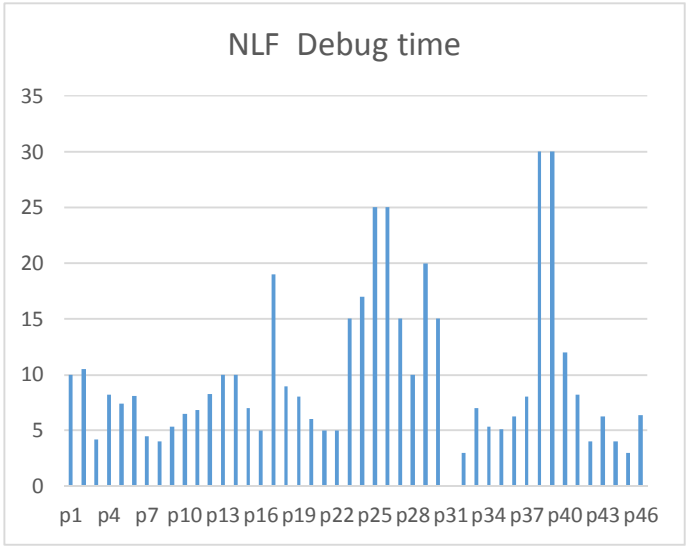
then novices in the conventional IDE group, however when they were asked to switch to NLF their performance enhanced both in grades and in debugging time , their overall debugging time improved in NLF due easy error messages and solution suggestions they strive less hard to correct errors than conventional environments used to write programs in FPL, performance and motivation was better and ensured deep learning then surface learning, with high self-efficacy, better understanding of errors and much improved degree of perseverance in FPL.

novice debugging time in conventional IDE verses in FPL Novices were given 45 programs , programs were to be written, debugged in conventional IDEs like Code Block, Dev, Turbo C and they were asked to write same programs in NLF, average time required to debug each program was less in NLF furthermore over all average debugging time was also less in NLF as compared to later, results are illustrated in following figure





The above figures clearly illustrates that novices took less time to debug in NLF based on CFG as compare to conventional IDEs with complex error messages. By the completion of FPL, novices in NLF performed better with elevated interest. Quality of programs produced by them was much better, with fewer errors and more readable. Average number of errors are much less. The T-test conducted on the performance of subjects shows that there was a significant difference on the score of conventional groups and NLF group such that **t-value = 1.67** and **p < .05**. It is concluded that over all NLF is useful, handy for majority of novices in CS1 in FPL. Through this research it is certain that natural

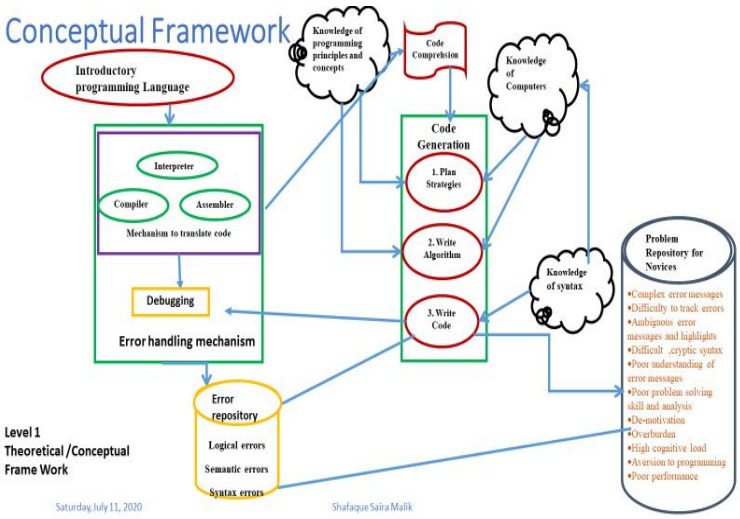


language based framework for handling errors in

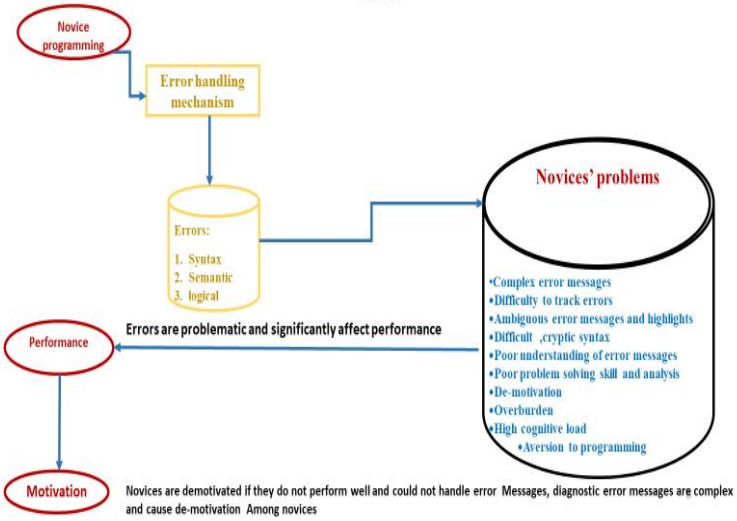
ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

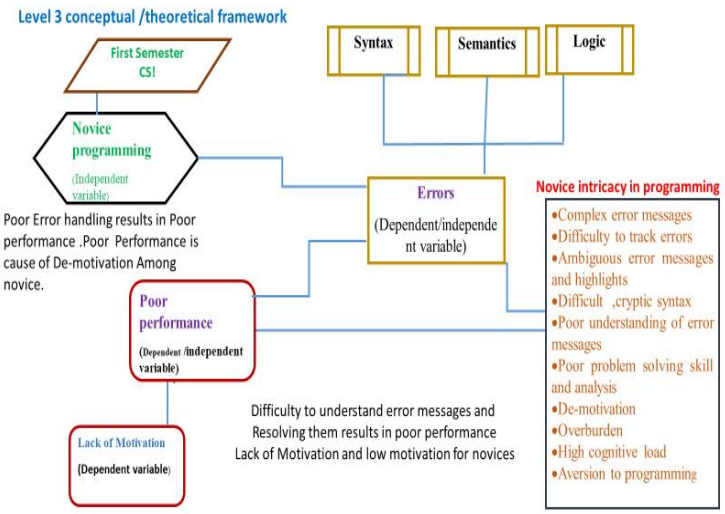
programming is specially advantageous for novice in their first semester as NLF comprehensively addressed many considerable facets which deters participation and progress of novices in computer science and programming .It is inferred that debugging becomes easy if error messages are in simple, easy to understand natural language then compiler jargons will ease leaning of programming with determination hence ensuring inventiveness, firmness and effective software development in computer science majors and encourages novices to trail their potential programming careers in CS

V. DISCUSSION



Level2: Figure 1.2 is viewed as conceptual framework for this research





The figures are viewed as conceptual framework of this research. The conceptual frame work of this research is to ensemble module of natural language as core component for error illustration and resolution in language modeling, which will significantly have correlation with performance and motivation of novice programmers. It is necessary to sure that the error messages should be novice friendly and represented in familiar vocabulary or showing hints then compiler jargons, complex terms and ambiguous sentences and is very important in designing error messages suggested by (Schliep, 2015). The syntax and semantic of any programming language have significant effect and the performance and motivation of students and as a result novice spent their most of the combating with the grammar of the programming language and could not develop skills like problem solving (Hooshyar, Alrashdan and Mikhak ,2013). (Ovsyannikov, M. K., & Kasimov, D. R. , 2014) concluded that multifaceted development environment and difficult syntax of the programming language elevate poor problem analysis and solving abilities for novice as a result of which novices less motivated, badly overstrained and continue to study introductory programming with repugnance and hence fight tough battle to learn syntax of the programming language.

The framework reflects the blueprint to ensemble language modeling through natural language error illustrator and resolver constituent to ease novice programmers for generating and writing bug free source code by implementing

ISSN: 2708-7123 | Volume-01, Issue Number-02 | June-2020
LC INTERNATIONAL JOURNAL OF STEM
Web: www.logicalcreations.org/stem | www.lcjstem.com | DOI: https://doi.org/10.47150

approach of enhanced compiler error messages. Figure. 1.1.1 represent conceptual frame work of this research and it is adapted from (M. Koorsse et al, 2014), snap-shooting code comprehension which is representing reliance relationship existing between diverse types of knowledge required by the neophyte programmer like syntax, programming principles, programming concept and skills required by novice programmers. Conceptual scaffold is illustration of language modeling programming paradigms like object oriented, imperative, procedural etc. In order to write program/source code each language is facilitated with blend of editors, integrated development environment (IDEs), graphical user interface (GUI). the only way to communicate with the machine is through compilers, interpreters, assemblers .The code generation phase is the most difficult and code writing extremely tricky and difficult as it is followed by complex language syntax and semantics and require lot of effort to write simple code accompanied by different types of errors (logical, semantic and syntax errors). For novice programmers writing source code and debugging is ordeal. Complex error messages are difficult to track. Precise programming knowledge and programming skills are required by novice programmers in order to write programs effectively (M. Koorsse et al, 2014). For novice programmer's inadequate compiler error messages are challenging and are primary assistance for debugging (Becker,2016). (Juded, 2005) reported that commercial compilers generate uninformative and sometimes miss leading error messages. Tracking errors is extremely hard for novice programmers. The debugging process and complier messages are source of high cognitive load, meager performance and motivation for novice programmers. Syntax understanding is thorny and solving syntax errors is somewhat very intricate process in programming. This conceptual model represents concept of inducing natural language to illustrate and resolve errors when writing source code, prior to compilation, conceptual model for my research represent concept of natural language programming assistance tools (PAT) to make program writing syntactically error free before compilation.

In 3rd figure conceptual frame work is represented with the set of independent and dependent variables and hence will have significant impact on this research. This researcher will focus on novices enrolled in first semester of computer science and novice programming is treated as independent variable, rest of the framework is dependent upon this sole variable. Novices encounter errors and it is dependent variable, hence errors messages, errors encountered, error resolution significantly influence performance of novices therefore errors are considered independent variable. Performance of novices is dependent upon errors therefore it is dependent variable. Performance directly influence motivation to learn programming, motivation is dependent upon performance and is dependent variable. Novice programming is influenced by the errors encountered and resolved, errors have impact on the performance of novices and performance has impact on motivation of novices.

variables of study

lists of dependent and independent variables:

Independent variables:	Dependent variables:
Novice programming	Errors
Error	Performance
Performance	Motivation

hypothesis:

H_0:

No co-relation exists between natural language design and impetus and performance of students/ novice programmers. Re-composition of compiler error messages by induction of natural language will have no significant impact on the performance and motivation of novice programmers in introductory programming courses.

H_1:

The performance, motivation and retention of novices are correlated with the induction of natural language in error resolution. Re-composition of compiler error messages in natural language has strong association with the performance of novices.

VI. CONCLUSION

The results of this study specify that NLF for elucidation of errors in programming increase learning and improve performance and inquisitiveness of novices, and that these escalations are of prim importance in order to enhance

performance and inspiration of novices reliably on fixing static errors.

NLF is effective tool in learning programming for novices and increase their skill in programming although and has profound outcome on the performance of both novices in FPL.NLF has significant impact on error resolution, diagnosis of errors, understanding of errors, identifying classes of errors in programming, effective debugging skills of novices is left for future work.

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 in the study.

REFERENCES

[1] Ali, A. &. (2014). Teaching an Introductory Programming Language. Journal of Information Technology Education: Innovations in Practice, 13, 57-67.

[2] Ali, A., & Shubra, C. (2010). Efforts to reverse the trend of enrollment decline in computer science programs: A case study. Issues in Informing Science and Information Technology, 7,209224.

[3] Ali, A., & Smith, D. (2014). Teaching an introductory programming language in a general education course. Journal of Information Technology Education: Innovations in Practice, 13, 57-67.

[4] Agrawal, S. K. (2016). Syntax errors identification from compiler error messages using ML techniques.

[5] Akcaoglu, M., & Koehler, M. J. (2014). Cognitive outcomes from the Game-Design and Learning (GDL) after-school program. Computers & Education, 75, 72-81.

[6] Andrzejewska, M., Stolińska, A., Błasiak, W., Pęczkowski, P., Rosiek, R., Rożek, B., ... & Wcisło, D. (2016). Eye-tracking verification of the strategy used to analyze algorithms expressed in a flowchart and pseudo code. *Interactive Learning Environments*, *24*(8), 1981-1995.

[7] Anderson, J. W., Tataru, P., Staines, J., Hein, J., & Lyngsø, R. (2012). Evolving stochastic context-free grammars for RNA secondary structure prediction. BMC bioinformatics, 13(1), 78.

[8] Bastani, O., Anand, S., & Aiken, A. (2015, January). Specification inference using context-free language reachability. In ACM SIGPLAN Notices (Vol. 50, No. 1, pp. 553-566). ACM.

[9] Benjamin S. Lerner, Matthew Flower, Dan Grossman, Craig Chambers, Searching for type-error messages, Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation, June 10-13, 2007, San Diego, California, USA

[10] Blok, T., & Fehnker, A. (2017). Automated Program Analysis for Novice Programmers. arXiv preprint arXiv:1710.00163.

[11] Biggers, M., Brauer, A., & Yilmaz, T. (2008, March). Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers. In ACM SIGCSE Bulletin (Vol. 40, No. 1, pp. 402-406). ACM.

[12] Brian W. Kernighan and Dennis M. Ritchie (1988), Section A13 of The C programming language, 2nd edition, by Prentice Hall.

[13] Bruner J. (1990). "Constructivist Theory." Retrieved 19 July, 2007, from http://tip.psychology.org/bruner.html.

[14] Becker, B. A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K., & Mooney, C. (2016). Effective compiler error message enhancement for novice programming students. Computer Science Education, 26(2-3), 148-175.

[15] Carter, J., & Jenkins, T. (2002). Gender differences in programming? Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education. Retrieved April 15, 2008 from ACM.

[16] C. Burrell and M. Melchert, "Augmenting compiler error reporting in the Karel++ microworld," Proceedings of the Conference of the National Advisory Committee on Computing Qualifications, 2007, p. 41--46.

[17] Carver, J. C., Henderson, L., He, L., Hodges, J., & Reese, D. (2007, July). Increased retention of early computer science and software engineering students using pair programming. In Software Engineering Education &

Training, 2007. CSEET'07. 20th Conference on (pp. 115-122). IEEE.

[18] Choi, H. (2012, August). Learners' reflections on computer programming using Scratch: Korean primary pre-service teachers' perspective. In 10th International Conference for Media in Education 2012 (ICoME) (pp. 22-24)

[19] Christian Murphy, Gail Kaiser, Kristin Loveland, Sahar Hasan, Retina: helping students and instructors based on observed programming activities, Proceedings of the 40th ACM technical symposium on Computer science education, March 04-07, 2009, Chattanooga, TN, USA

[20] Clark, D., MacNish, C. & Royle, G.F (1998). Java as a teaching language--opportunities, pitfalls and solutions. The proceedings of the third Australasian conference on computer science education (July 1998), ACM Press, 173-179.

[21] Cindy Norris, Frank Barry, James B. Fenwick Jr., Kathryn Reid, Josh Rountree, Clock It: collecting quantitative data on how beginning software developers really work, ACM SIGCSE Bulletin, v.40 n.3, September 2008

[22] Cohoon, J. (2006) Just get over it or get on with it: Retaining women in undergraduate computing. Women and Information Technology 205--237

[23] Cohen, J. A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1):37--46, 1960.

[24] Coull, N.J. SNOOPIE: Development of A Learning Support Tool for Novice Programmers Within A Conceptual Framework. PhD Thesis, School of Computer Science, University of St. Andrews, 2008.

[25] Danial Hooshyar, Moslem Yousefi and Heuiseok Lim, A systematic review of data-driven approaches in player modeling of educational games, Artificial Intelligence Review, (2017)

[26] Dann, W., Copper, S., & Pausch, R. (2006). Learning to program with Alice. Upper Saddle River, NJ: Prentice Hall.

[27] Daly, T. (2011, May). Minimizing to maximize: An initial attempt at teaching introductory programming using Alice. Journal of Computer Science in Colleges, 26(5), 23-3

[28] Denny et al. (2014). Code write :dupporting students driven practices of java. ACM , 471-476.

[29] Denny, P., Luxton-Reilly, A., & Carpenter. . (2014). Enhancing syntax error messages appears ineffectual. ACM , 273-278.

[30] Denny, P., Luxton-Reilly, A., & Carpenter, D. (2014, June). Enhancing syntax error messages appears ineffectual. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 273-278). ACM.

[31] de Raadt, M., Watson, R., & Toleman, M. (2003). Introductory programming languages at Australian universities at the beginning of the twenty first century. Journal of Research and Practice in Information Technology, 35(3), 163.

[32] de Raadt, M., Hamilton, M., Lister, R. F., Tutty, J., Baker, B., Box, I., ... & Petre, M. (2005). Approaches to learning in computer programming students and their effect on success. Research and Development in Higher Education Series

[33] de Raadt, M., Toleman, M., & Watson, R. (2004). Training strategic problem solvers. ACM SIGCSE Bulletin, 36(2), 48-51.

[34] Elliot Soloway , James C. Spohrer , , Novice mistakes: are the folk wisdoms correct?, Communications of the ACM, v.29 n.7, p.624-632, July 1986

[35] Engelfriet, J. (2014). Context-free grammars with storage. arXiv preprint arXiv:1408.0683.

[36] Essi Lahtinen, Kirsti Ala-Mutka, Hannu-Matti Järvinen, A study of the difficulties of novice programmers, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, June 27-29, 2005, Caparica, Portugal

[37] E. Soloway, James C. Spohrer, Studying the Novice Programmer, L. Erlbaum Associates Inc., Hillsdale, NJ, 1988.

[38] Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., de Raadt, M., ... & Petre, M. (2006, January). Predictors of success in a first programming course. In Proceedings of

the 8th Australasian Conference on Computing Education-Volume 52 (pp. 189-196). Australian Computer Society, Inc.

[39] Findler, R. B. (2010). DrRacket: Programming Environment.

[40] Freeman, S. F., Jaeger, B. K., & Brougham, J. C. (2004). Pair programming: More learning and less anxiety in a first programming course. *age*, *8*, 1.

[41] Gordon, V. N., & Steele, G. E. (2003). Undecided first-year students: A 25-year longitudinal study. Journal of the First-Year Experience & Students in Transition, 15(1), 19-38Michael de Raadt, R. W. (2002, june). Language Trends in Introductory Programming Courses. Informing Science InSITE - "Where Parallels Intersect" June 2002.

[42] Gross, P. A., Herstand, M. S., Hodges, J. W., & Kelleher, C. L. (2010, February). A code reuse interface for non-programmer middle school students. In Proceedings of the 15th international conference on Intelligent user interfaces (pp. 219-228). ACM.

[43] Hagan, D. and Markham, S. Teaching Java with the BlueJ environment. In Proceedings of Australasian Society for Computers in Learning in Tertiary Education Conference. Citeseer, 2000.

[44] Hage, J. and Keeken, P.V. Mining Helium programs with Neon. In Technical Report, Department of Information and Computing Sciences, Utrecht

[45] Hooshyar, D., Maíen, T., & Masih, M. (2013). Flowchart-based programming environments aimed at novices. International Journal of Innovative Ideas, 13(1), 52-62.

[46] Herbert, C. (2007). An introduction to programming with Alice. Boston, Massachusetts: Course Technology

[47] Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. J. (2015). A flowchart□based intelligent tutoring system for improving problem□solving skills of novice programmers. Journal of Computer Assisted Learning, 31(4), 345-36.

[48] Iain Milne, Glenn Rowe, Difficulties in Learning and Teaching Programming—Views of Students and Tutors, Education and Information Technologies, v.7 n.1, p.55-66, March 2002

[49] Isong, B. (2014). A Methodology for Teaching Computer Programming: first year students' perspective. International Journal of Modern Education and Computer Science, 6(9), 15.

[50] Isong, I. A., Rao, S. R., Holifield, C., Iannuzzi, D., Hanson, E., Ware, J., & Nelson, L. P. (2014). Addressing dental fear in children with autism spectrum disorders: a randomized controlled pilot study using electronic screen media. Clinical pediatrics, 53(3), 230-237.

[51] Isa, B. S., Boyle, J. M., Neal, A. S., & Simons, R. M. (1983, December). A methodology for objectively evaluating error messages. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 68-71). ACM.

[52] Jackson, J., Cobb, M., and Carver, C. Identifying top Java errors for novice programmers. In Proceedings of the Frontiers in Education Conference, pages T4C--24. 2005.

[53] Jadud, M.C. A First Look at Novice Compilation Behaviour Using BlueJ. Computer Science Education, 15(1):25--40, 2005

[54] Munson, J. P., & Zitovsky, J. P. (2018, February). Models for Early Identification of Struggling Novice Programmers. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 699-704). ACM.

[55] J.S. Rey, From Alice to BlueJ: a transition to Java, Master's thesis, School of Computing, Robert Gordon University, 2009.

[56] Kamada, M. (2016, November). Islay—An educational programming tool based on state diagrams. In Advances in Electrical, Electronic and Systems Engineering (ICAEES), International Conference on (pp. 230-232). IEEE.

[57] Kiezun, A., Ganesh, V., Guo, P. J., Hooimeijer, P., & Ernst, M. D. (2009, July). HAMPI: a solver for string constraints. In Proceedings of the eighteenth international symposium on Software testing and analysis (pp. 105-116). ACM.

[58] Köksal, M.F., Baar, R.E., and Üsküdarl1, S. Screen-Replay: A Session Recording and Analysis Tool for

DrScheme. In Proceedings of the Scheme and Functional Programming Workshop, Technical Report, California Polytechnic State University, CPSLO-CSC-09-03, pages 103--110. 2009.

[59] Kölling and McCall, D., M. 2014. Meaningful Categorisation of Novice Programmer Errors. Frontiers in Education Conference (2014), 2589--2596.

[60] Kummerfeld, S. K., and Kay, J. The neglected battle fields of syntax errors. In Proceedings of the Fifth Australasian Conference on Computing Education -Volume 20 (Darlinghurst, Australia, Australia, 2003), ACE '03, Australian Computer Society, Inc., pp. 105–111

[61] Kyfonidis, C., Moumoutzis, N., & Christodoulakis, S. (2017, April). Block-C: a block-based programming teaching tool to facilitate introductory C programming courses. In Global Engineering Education Conference (EDUCON), 2017 IEEE (pp. 570-579). IEEE.

[62] Kyfonidis, C., Moumoutzis, N., & Christodoulakis, S. (2015). Block-c: A block-based visual environment for supporting the teaching of c programming language to novices. Google Scholar.

[63] Laurie Murphy, Gary Lewandowski, Renée McCauley, Beth Simon, Lynda Thomas, Carol Zander, Debugging: the good, the bad, and the quirky -- a qualitative analysis of novices' strategies, Proceedings of the 39th SIGCSE technical symposium on Computer science education, March 12-15, 2008, Portland, OR, USA.

[64] M.C. Jadud, "A First Look at Novice Compilation Behaviour Using BlueJ," Computer Science Education, vol. 15, Mar. 2005, p. 25--40.

[65] Malik, S. I., & Coldwell-Neilson, J. (2017). A model for teaching an introductory programming course using ADRI. Education and Information Technologies, 22(3), 1089-1120.

[66] Marceau, G., Fisler, K., & Krishnamurthi, S. (2011, October). Mind your language: on novices' interactions with error messages. In Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software (pp. 3-18). ACM.

[67] Marceau, G., Fisler, K., & Krishnamurthi, S. (2011, March). Measuring the effectiveness of error messages designed for novice programmers. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 499-504). ACM.

[68] Mendelson, P., Green, T. R. G. and Brna, P. (1990) Programming languages in education:the search for an easy start. In J.-M. Hoc, T. R. G. Green, D. Gilmore and R. Samway(eds) Psychology of Programming, pp. 175–200, London; Academic Press.

[69] M.M. Ben-Ari, "Compile and Runtime Errors in Java," http://stwww.weizmann.ac.il/g-cs/benari/oop/errors.pdf, accessed June 15, 2010.

[70] Mason, R., & Cooper, G. (2014, January). Introductory Programming Courses in Australia and New Zealand in 2013-trends and reasons. In Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148 (pp. 139-147). Australian Computer Society, Inc.

[71] Matsuzawa, Y., Ohata, T., Sugiura, M., & Sakai, S. (2015, February). Language migration in non-cs introductory programming through mutual language translation environment. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (pp. 185-190). ACM.

[72] Matthew C. Jadud, Methods and tools for exploring novice compilation behavior, Proceedings of the second international workshop on Computing education research, September 09-10, 2006, Canterbury, United Kingdom

[73] Mathew, B. d. (1984). Fatal error in Pass Zero:how not to confuse novices. Behaviour and Information Teachnology , 109-118.

[74] . Marie-Hélène Nienaltowski , Michela Pedroni , Bertrand Meyer, Compiler error messages: what can help novices?, Proceedings of the 39th SIGCSE technical symposium on Computer science education, March 12-15, 2008, Portland, OR, USA

[75] Maria Hristova, Ananya Misra, Megan Rutter, Rebecca Mercuri, Identifying and correcting Java programming errors for introductory computer science students, Proceedings of the 34th SIGCSE technical symposium on

Computer science education, February 19-23, 2003, Reno, Navada, USA

[76] Mason, R., & Cooper, G. (2012, January). Why the bottom 10% just can't do it: mental effort measures and implication for introductory programming courses. In Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123 (pp. 187-196). Australian Computer Society, Inc.

[77] Mohamed Shuhidan, S., Hamilton, M., & D'Souza, D. (2011, June). Understanding novice programmer difficulties via guided learning. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (pp. 213-217). ACM.

[78] Montesi, F., Guidi, C., & Zavattaro, G. (2014). Service-oriented programming with jolie. Web Services Foundations, 81-107.

[79] Munson, J. P., & Schilling, E. A. (2016). Analyzing novice programmers' response to compiler error messages. Journal of Computing Sciences in Colleges, 31(3), 53-61.

[80] Murphy, L., Lewandowski, G., McCauley, R., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: the good, the bad, and the quirky--a qualitative analysis of novices' strategies. ACM SIGCSE Bulletin, 40(1), 163-167.

[81] Nakamura, S., Nozaki, K., Morimoto, Y., & Miyadera, Y. (2014, September). Sequential pattern mining method for analysis of programming learning history based on the learning process. In Education Technologies and Computers (ICETC), 2014 The International Conference on (pp. 55-60). IEEE

[82] Naveed, S., Sarim, M., & Nadeem, A. (2018). C in CS1: Snags and viable solution. Mehran University Research Journal of Engineering & Technology, 37(1), 1.

[83] Naveed, M. S., Sarim, M., & Ahsan, K. (2016). Learners programming language a helping system for introductory programming courses. Mehran University Research Journal of Engineering & Technology, 35(3), 347.

[84] Naveed, M. S., Sarim, M., & Nadeem, A. Making C a Primary Language for the First Programming Course.

[85] Nelson Laird, T. F., & Garver, A. K. (2010). The effect of teaching general education courses on deep approaches to learning: How disciplinary context matters. Research in Higher Education, 51(3), 248-265.doi:10.1007/s11162-009-9154-7

[86] N.J. Coull, SNOOPIE: development of a learning support tool for novice programmers within a conceptual framework, PhD Thesis, School of Computer Science, University of St. Andrews, 2008.

[87] N. Wirth, "The Programming Language Pascal," Acta Informatica, vol. 1, 1971, pp. 35-63.

[88] Ovsyannikov, M. K., & Kasimov, D. R. (2014). Editor and Interpreter of Program Flowcharts for Distance Learning Programming. *Bulletin of Kalashnikov ISTU*, (3), 154-156.

[89] Ozoran, D., Cagiltay, N., & Topalli, D. (2012). Using scratch in introduction to programming course for engineering students. In 2nd International Engineering Education Conference (IEEC2012) (pp. 125-132).

[90] Paul Gross, Kris Powers, Evaluating assessments of novice programming environments, Proceedings of the first international workshop on Computing education research, p.99-110, October 01-02, 2005, Seattle, WA, USA University.

[91] Pham, B. (1996). The changing curriculum of computing and information technology in Australia. Proceedings of the second Australasian conference on computer science education (July 1996), ACM Press, 149-154.

[92] Piteira, M., & Costa, C. (2013, July). Learning computer programming: study of difficulties in learning programming. In Proceedings of the 2013 International Conference on Information Systems and Design of Communication (pp. 75-80). ACM.

[93] Piteira, M., & Costa, C. (2012, June). Computer programming and novice programmers. In Proceedings of the Workshop on Information Systems and Design of Communication (pp. 51-53). ACM.

[94] Porter, R., & Calder, P. (2004). Patterns in learning to program: an experiment? Proceedings of the Sixth Conference on Australasian Computing Education – Volume 30, 241 -246. Retrieved April 18, 2008 from ACM.

[95] Powers, K., Ecott, S., & Hirshfield, L. (2007). Through the looking glass: Teaching CS0 with Alice. ACM

SIGCSE Bulletin, 39(1), 213-217. Retrieved March 28, 2008 from ACM.

[96] Raina Mason and Simon the 20th Australasian Computing Education Conference ACE '18 Brisbane, Queensland, Australia Proceedings of the 20th Australasian Computing Education Conference on - ACE '18 ACM Press New York, New York, USA, (2018).

[97] Rohrmeier, M., Fu, Q., & Dienes, Z. (2012). Implicit learning of recursive context-free grammars. PloS one, 7(10), e45885.

[98] Robert Bruce Findler, John Clements, Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Paul Steckler, Matthias Felleisen, DrScheme: a programming environment for Scheme, Journal of Functional Programming, v.12 n.2, p.159-182, March 2002

[99] Robins A, Rountree J, et al. (2003). "Learning and Teaching Programming: A Review and Discussion." Journal of Computer Science Education 13(2): 137--172

[100] Schliep, P. A. (2015). Usability of Error Messages for Introductory Students. Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal, 2(2), 5.

[101] Shapiro, R. B., & Ahrens, M. (2016). Beyond blocks: Syntax and semantics. Communications of the ACM, 59(5), 39-41.

[102] Seymour, E. and Hewitt, N. (1997). Talking about leaving: Why undergraduates leave the sciences. Boulder, CO: Westview PressRonit Ben-Bassat Levy , Mordechai Ben-Ari , Pekka A. Uronen, The Jeliot 2000 program animation system, Computers & Education, v.40 n.1, p.1-15, January 2003 .

[103] Singh, R., Gulwani, S., & Solar-Lezama, A. (2013). Automated feedback generation for introductory programming assignments. ACM SIGPLAN Notices, 48(6), 15-26.

[104] Smith, G., & Fidge, C. (2008, January). On the efficacy of prerecorded lectures for teaching introductory programming. In Proceedings of the tenth conference on Australasian computing education-Volume 78 (pp. 129-136). Australian Computer Society, Inc.

[105] Striewe, M., & Goedicke, M. (2014, June). A review of static analysis approaches for programming exercises. In International Computer Assisted Assessment Conference (pp. 100-113). Springer, Cham.

[106] Taheri, S. M., Sasaki, M., & Ngetha, H. T. (2015, July). Evaluating the effectiveness of problem-solving techniques and tools in programming. In Science and Information Conference (SAI), 2015 (pp. 928-932). IEEE.

Teague, D., & Roe, P. (2008, January). Collaborative learning: