SOFTWARE BUGS, DETECTION, ANALYSIS AND FIXING IN COMPUTER LABORATORY SYSTEM OF EBONYI STATE COLLEGE OF EDUCATION, IKWO.

Chima, Agwuama Okporie
Department Of Computer Science
Ebonyi State College Of Education, Ikwo
Chimaagwuama0@Gmail.Com

Anikeze, Christiana Obiageriaku
Department Of Computer Science
Ebonyi State College Of Education, Ikwo
Get2christyanikeze@Gmail.Com

Ekechi, Jeffrey Chikwado
Department Of Computer Science
Ebonyi State College Of Education, Ikwo
Kwadojeff6@Gmail.Com

Nwakpa, Matthew Nwankwo
Department Of Computer Science
Ebonyi State College Of Education, Ikwo
Monnon200@Gmail.Com

Abstract

This research is meant to determine the software bugs, detection, analysis and fixing, hence the research will help to produce high quality software to satisfy a set of functional and non functional requirements. Besides, this study is carried out at the Computer Laboratory of the Ebonyi State College of Education, Ikwo, where the software has transformed the ways through which many applications are done, whereby the research acknowledges that software is a problem solving activity. The software has ensures transparency of deep learning knowledge and enabling human involvement in decision making. The author discusses on the research questions and hypothesis that guided the study and was tested at 0.05 level of significance. To foster this acceptance, the paper demonstrated the analysis of the table where the four rating scale of strongly agree, agree, disagree and strongly disagree was examined. The paper explains that effective software bugs leads to goal accomplishment at a minimum resources, and therefore calls for new orientations in handling software devices vis-a-vis the use of analysis and synthesis.

Keyword: software, bug tracker, laboratory, developers.

Introduction

The origin of software dates back to the mid-20th century, when the first computers were developed.

According to (Anatutu, 2021), he highlighted the origin of software to include, the early beginning's from 1940s - 1950s, first programming languages from 1950s - 1960s, software industry emerges from 1960s - 1980s and modern software development from 1990s - Present.

Eze (2015), described software as programs resident in the memory of the computer system as at the time of purchase. Software has become a key feature of a rapidly growing range of products and services from all sectors of economic activity. Software intensive systems include large-scale heterogeneous systems, embedded systems for automotive applications, telecommunications, wireless ad-hoc systems, business applications with an emphasis on web services, etc. Our daily lives depend on complex software-intensive systems, from banking to communications to transportation and even medicine. Software technology is a driving factor for many high-tech products; competence in software technology defines more and more the innovative capabilities of industries. However, in the Computer Science Laboratory of the Ebonyi State College of Education, Ikwo, one may argue that software developed in most organizations today seldom is of high quality and unstable. There is always a communication gap between software developers and end-users of a software system; this leads the software being abandoned. Also due to this communication gap the software developers hardly get feedbacks from users of the systems and the few ones they get are hardly properly documented (Kadir, 2016). There is also lack of software bug data repository which makes it difficult to make effective decisions on the improvement of the software development process.

To compensate for this lack of quality and stability, software developing organizations embraced four common approaches towards improving it. The first is to hire the best and brightest personnel to develop bug free software, although the criteria for selecting such personnel are hardly ever defined and it is quite impossible for an individual to develop a bug free system that meets the expectations of several users (Gerald, 2017). Another is to reuse software instead of developing it anew. Unfortunately, few organizations have been able to develop general, reliable software that can be reused without significant modification. Yet another scheme is to develop software at higher levels of abstractions. However, it is still rare to convince others of the wisdom of this approach in light of anticipated system performance penalties.

Statement of problem

The statement of the problem include technical Problems, user-related problems, maintenance and support problems, economic and social problems and environmental problems

Objectives

The objectives of this study is to develop an application that would:

- ❖ Enable users and testers to report bugs
- ❖ Bridge the communication gap between developers, testers and end-users of software systems
- ❖ Provide a software bug repository for researchers and developers

Significance

The significance include

- ❖ It helps to identify cost effective- methods of developing and managing software bugs
- ❖ The combined benefit of such a software bug detection methodology would be significant because defect detection and correction activities consume about fifty percent of the labour to create software systems and as much as seventy-five percent of the total software's life cycle costs
- ❖ This would lead to an increase in profit made from systems as there would be a reduction in costs incurred as a result of bugs in software systems.
- ❖ The bug tracker to be developed at the end of this research would serve as bug data repository which can be used both by developers and software quality researchers to track bugs, predict and correct potential bugs before they manifest thereby increasing productivity.
- ❖ To the society it provides product reliability information to potential buyers of the software system thereby increasing the confidence of users in a software product.

Limitations

1. The major problem here is lack of comprehensive software development data.

2. Many software users are interested on how the software run and not how it was produced.
3. In addition to the limitations, some software developed do not meet the target of developing them

Software Bug

Software bugs are unexpected outcomes or errors in computer software that manifest as undesired behavior, crashes, or incorrect output and stem from errors in the software development process during design, coding, or testing. In addition, they could be seen as faults, flaws, or errors in computer software that result in unexpected or unanticipated outcomes (Jaque, 2017).

Fink (2018), pointed out various ways through which software could be identified to include, undesired behavior, system crashes, freezes or erroneous and insufficient output.

These issues result from errors created in the software development process when the software is being designed, coded, or tested.

How To Avoid Software Bug. Agwu (2019), mentioned various ways through which software bugs could be avoided

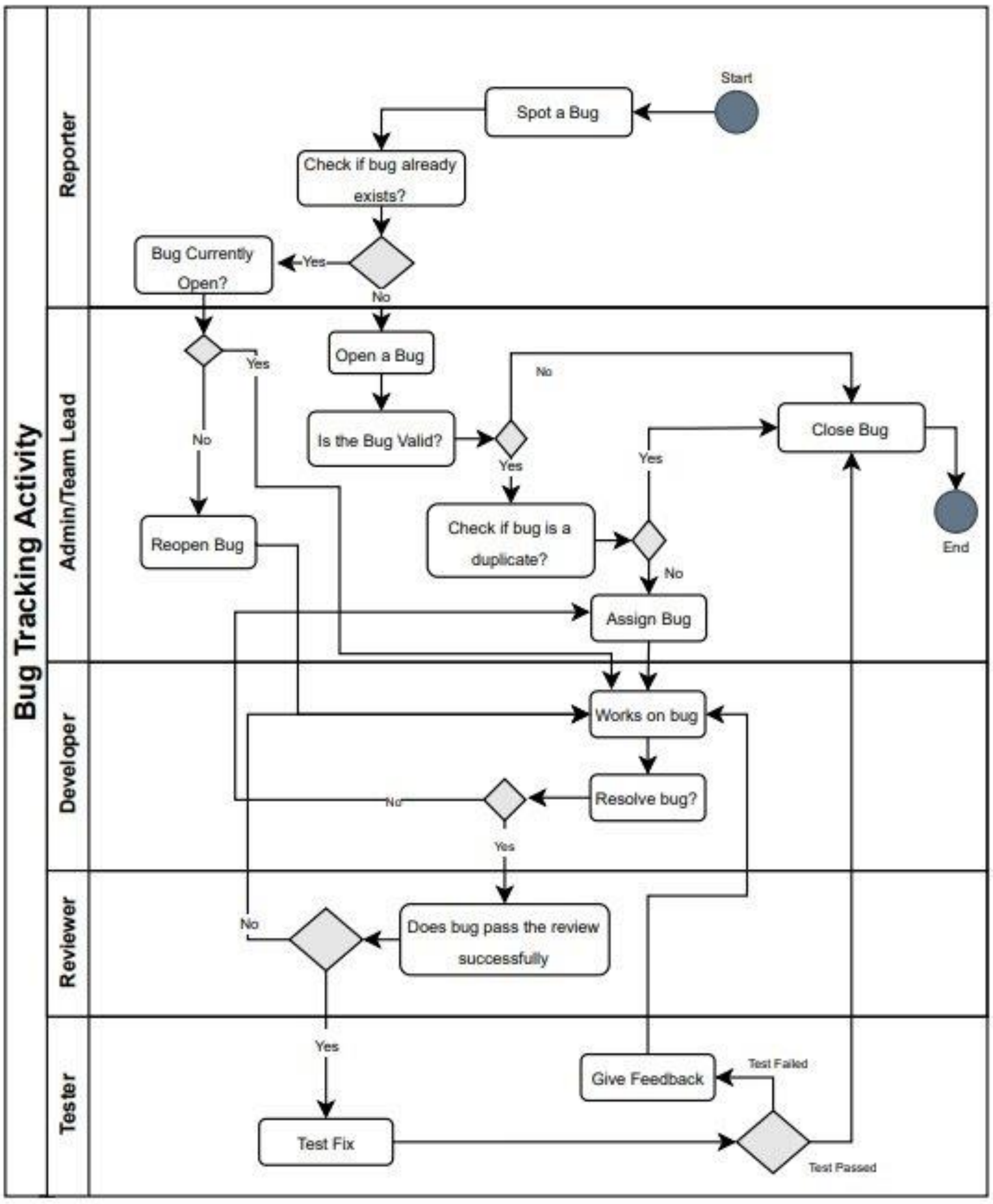
1. A comprehensive strategy that takes into account different phases of the software development lifecycle is necessary to prevent software bugs. Defining precise and thorough requirements and specifications ensures that developers know exactly what the program is meant to do and helps to eliminate confusion
2. In-depth planning and design stages let developers see possible problems and create reliable, scalable solutions
3. Following recommended techniques for coding, such as modularization, appropriate documentation, and standard naming conventions, can improve the readability and maintainability of code and lower the risk of errors being introduced.
4. Thorough testing methods, such as system integration and unit testing, aid in the early detection of defects in software development, enabling appropriate adjustments.

5. Continuous integration and deployment practices allow regular code integration and automated testing which speeds up the identification and fixes of bugs.

6. Development teams can improve overall code quality and lower the frequency of problems by promoting a culture of teamwork, peer review, and information sharing.

7. Post-deployment reporting and monitoring systems allow developers to quickly resolve any unexpected problems that crop up during real-world use, guaranteeing that software is durable and dependable over time.

Flow Chat of Software Bug Activities



Bug Tracking Activity

Software Bug Data

Software bug data has to do with the information collected and stored about software bugs, including their characteristics, behavior, and resolution. This data can be used to improve software quality, reduce debugging time, and enhance the overall software development process.

Various Software Bug Data include, (Ibrahim, 2022)

1. Bug Reports: Detailed descriptions of bugs, including steps to reproduce, expected results, and actual results.
2. Bug Tracking Data: Information about bug status, priority, severity, and assignment to developers.
3. Code Review Data: Feedback and comments from code reviews, which can help identify potential bugs.

4. Test Results: Data from automated tests, including pass/fail rates, error messages, and test coverage.

5. Crash Dump Data: Information about software crashes, including error messages, stack traces, and system configuration.

Software Analysis

Software analysis involves the process of examining software systems to understand their behavior, structure, and performance. It involves using various techniques and tools to analyze the software's source code, design, and functionality.

Challenges in Software Analysis. Kalu and Nkama (2017)**,**

1. Complexity: Analyzing complex software systems with many interconnected components.
2. Scalability: Analyzing large software systems with millions of lines of code.
3. Accuracy: Ensuring the accuracy of analysis results, which can be affected by various factors, including tool limitations and human error.

Software Testing

Software testing ensures the processes of evaluating a software application or system to ensure it meets the required specifications, works as expected, and is free from defects.

Challenges in Software Testing. According to Ayodele (2019), he mentioned the challenges of software testing to include

1. Time Constraints: Testing within tight deadlines and budgets.
2. Complexity: Testing complex software systems with many interconnected components.
3. Limited Resources: Dealing with limited resources, including personnel, equipment, and budget.
4. Constant Change: Adapting to changing software requirements and technologies

Software Detection

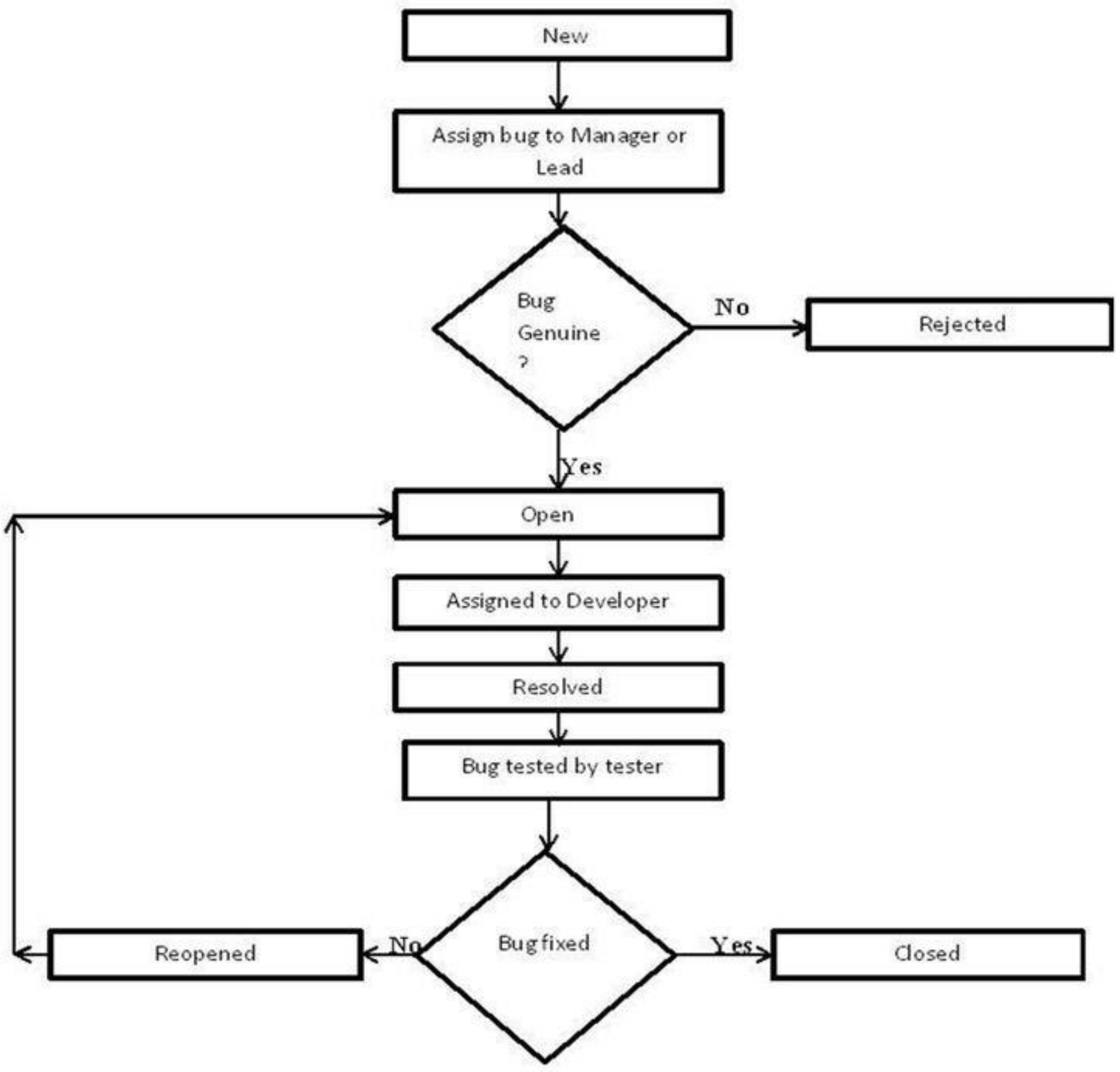
Software detection refers to the process of identifying and recognizing software applications, components, or libraries within a system, network, or device. This can be done for various purposes, including:

Software Fixing

Amarachi and Nnachi (2023), defined software fixing as the process of identifying, analyzing, and resolving defects or bugs in software applications. This process involves several steps:

Steps in Software Fixing

1. Bug Reporting: Identifying and reporting bugs or defects in the software.
2. Bug Triage: Evaluating and prioritizing bugs based on their severity and impact.
3. Bug Analysis: Analyzing the bug to identify its root cause and potential solutions.
4. Code Changes: Making changes to the code to fix the bug.
5. Testing: Verifying that the bug is fixed and that the changes did not introduce new bugs.
6. Deployment: Deploying the fixed software to production.



Bug Life Cycle in Software Testing

Research Questions

1. What are the mean on how software problems could be bug?
2. What are the mean on how software problems could be detected?
3. What are the mean on how software problems could be tested?
4. What are the mean on how software problems could be fixed?

Hypothesis

1. There is no significant difference between the mean software programs could be bug.
2. There is no significant difference between the mean software programs could be detected.
3. There is no significant difference between the mean software programs could be tested.
4. There is no significant difference between the mean software programs could be fixed.

Analysis of the Results

The analysis of results using four rating scales of Strongly Agree, Agree, Strongly Disagree and Disagree

Table 1:

Mean ratings and standard deviation on Software Bugs Tracker

S/N	Statement	SA	A	D	SD	Mean	SD	Decision
1.	Software development process	23	55	56	78	2.11	1.03	Disagree
2.	Software sample runs	62	58	49	43	2.66	1.11	Agree
3.	Software administrative module	31	44	49	88	2.08	1.10	Disagree
4.	Software product reliability	72	65	39	36	2.82	1.03	Agree
5.	Software tester and user module	87	61	42	22	3.00	1.01	Agree
6.	Software report bug	89	61	52	10	3.08	0.92	Agree
7.	Software edit bug	91	85	33	3	3.25	0.76	Agree
8.	Software system stability	26	52	66	68	2.17	1.02	Disagree
9.	Software development	78	54	52	28	2.86	1.06	Agree
10.	Software system flowcharting	89	71	34	18	3.09	0.96	Agree

N = 212

Table 1:

Table 1 shows that the means ratings of the ten items and the criterion mean rate of 2.50, that is in agreement and disagreement on software bug tracker of the laboratory system

Table 2:

Mean ratings and standard deviation on software system testing

S/N	Statements	SA	A	D	SD	Mean	SD	Decision
1.	Database management system	21	19	15	12	2.73	1.10	Agree
2.	Data flow analysis	33	21	7	6	3.21	0.96	Agree
3.	Unit testing	29	19	11	8	3.03	1.04	Agree
4.	Integrated testing	41	11	8	7	3.28	1.04	Agree
5.	Design specifications	1	5	22	39	1.52	0.70	Disagree
6.	Unified testing methodology	7	13	19	28	1.99	1.02	Disagree
7.	Cost and benefit analysis	31	20	9	7	2.66	0.66	Agree
8.	Current bug classifications	25	21	19	2	3.03	0.89	Agree
9.	Software management bugs	24	19	13	11	2.83	1.10	Agree
10.	PHP hypertext programming language	29	19	15	4	3.09	0.95	Agree

N = 67

Table 2:

Table 2 shows the mean ratings of the ten items shows and the criterion mean rate of 2.50, in agreement and disagreement of software system testing in the laboratory system.

Table 3:

Mean ratings and standard deviations on software system documentations

S/N	Statement	SA	A	D	SD	Mean	SD	Decision
1.	internal documentations	87	31	19	5	3.41	0.82	Agree
2.	external documentations	7	14	30	91	1.52	0.86	Disagree
3.	tracker source code	14	30	38	60	1.99	1.02	Disagree
4.	system functionality	21	27	43	51	2.13	1.06	Disagree
5.	documentations and modifications	60	38	30	14	3.01	1.02	Agree
6.	structural query language	19	23	37	63	1.99	1.07	Disagree
7.	hyper text make up language	17	21	41	63	1.94	1.04	Disagree
8.	design stage	63	41	21	17	3.06	1.04	Agree
9.	coding and implementations	21	33	38	50	2.17	1.07	Disagree
10.	documentations and implementation	91	30	14	7	3.44	0.86	Agree

N = 142

Table 3:

Table 3 shows the mean ratings of the ten items and the criterion mean value of 2.50, that is in agreement and not in agreement on software system documentations

SUMMARY OF ACHIEVEMENTS

This research work identified a post-deployment method of managing bugs in software system that would increase software stability and reliability while at the same time reduce the cost of development and management. A web based application called Bug Tracker was later development to track bugs in developed software system and serve as a software bug repository.

This application helps to track bugs in software 6th system and makes their removal easier.

It serves as a software bug data repository for researchers on software quality, which can be used to optimize software development process and also identify and correct possible future bugs before they manifest.

AREAS OF APPLICATION OF WORK

This research work is applicable to students or researchers in the field of software engineering, software development, and software quality among a host of other areas of research that is related to software development. At the same time, this research work is very relevant to developers of software system whose aim is to ensure the reliability and quality of developed software system.

It provides insight into how software can be managed especially during testing and post-deployment.

SUGGESTIONS FOR FURTHER WORK

There are several aspects of this research that can be pursued.

One of such would be to conduct a cost and benefit analysis of current bug classification with a view of improving the current ones. This could help to facilitate more comprehensive bug reports and effective bug analysis which would definitely improve the process of bug removal and overall software reliability while further reducing cost.

RECOMMENDATIONS

The Bug tracker application developed in this research provides a typical implementation of a functional Bug Tracking system to manage software testing and post-deployment maintenance. Therefore we would recommend organizations and academia should study the nature of bugs as they occur and are reported via the Bug tracker, in other to create better means of preventing and handling bugs

It is also recommended that the academia intensify research into the development of a unified software testing methodology.

There’s also a need to create more awareness for software developers on the advantage of software testing and bug reporting. This would increase the general acceptability and popularity of their software product especially for software developers in Nigeria.

CONCLUSION

The main findings of this research work is that it's possible to increase a software system's stability, reduce the cost of software development especially after deployment and also increase the confidence of users in a software system if a well-planned post-deployment strategy is implemented.

This was demonstrated through the use of Bug tracker as it helped to manage the software development and maintenance of Brain Bench Technologies and Diva Soft, while at the same time provided a bug-data repository which served as a rich source of data, when they were taking decisions on ways of improving their software development process.

The back and forth interaction that is logged and made available to users of the Bug Tracker helped both the developers and software testers to understand the software's operations—control and data flow—responsible for failures, which on the long term increased the stability and reliability of developed software, this lead to an increased customer satisfaction and reduced operational cost for the software development organization.

REFERENCES

Agwu, S.N. (2019). The amalgamation of the information technology in the computer world. Abba Nna Printing Company. Volume 2, 8-12

Amarachi, S.L. and Nnachi, U.Z (2023). Information and communication technology in the 21st century. A lecture delivered at the 20th International Conference of Computer Society of Nigeria.

Anatutu, J.O (2021). Availability of ICT facilities in science and technology, Awka, Anambra State Publishing Firm.

Ayodele, K.T (2019). The application of computer science education in the tertiary institutions. Nwantinti Printing Press, Enugu.

Eze, D.O (2015). The relationship between availability of technological equipment and student's performance in Afikpo Education Zone. Unpublished MEd. Thesis, Department of Vocational Technical Education, U.N.N

Fink, D.C (2018). Guideline for the successful adoption of information technology in small and medium enterprises. International journal of communication science. 22, 262-274

Gerald, C. (2017). Modern Techniques in Computer Simulation. www.amazonbooks.org

Ibrahim, S.J. (2022). New approaches in communication. A compendium of papers presented at the 24th Computer Association of Nigeria (COAN), held at Akure, June 7 – 15

Jaque, R. (2017). Computer Added Instructions. A paradigm Shift International Journal of Computer Science, 5(1), 610 - 627. http; www. LICS ICT.edu.org

Kadir, D. (2016). A Rationale for using Computer in Science Education. The American Computer Teacher, Volume 1, (46), pp 200 - 216. www.ACT.teach.net

Kalu, E.G. and Nkama, T.O. (2017). Computer calculated games and arithmetic in the 21st century. Chimamaka Pringing Technology, Owerri.

Leghara, B.N. and Mbah, N.C (2019). Competencies skills needed by STM teacher towards the development of entire prenuerial skills I students proceedings of the 50th annual conference of STAN 31 - 37