Locking Editor: A Utility For Protecting Software Exercises In The Computer Laboratory Of AMA University

Paul M. Grafilon, Jonathan Fuller, Shiitj B. Mer, Tristan Jay P. Calaguas

Abstract: The student of AMA University persistence in computing which has the keys to providing their talent needed to fill the computer laboratory in the computing professions. A range of factors can affect a student's decision to remain in a computing major or change to another major if ever they feel that computing education is difficult. This has to describe the activities in computer laboratory specifically exercises, machine problems, and computing case studies interacting different application programs as the basis of their skills and knowledge in programming capability. The nature of those activities addresses by using of IDE as open source in all programming applications which may result of specific intervention such as using the editor to create a source file, the code blocks, comments, and program statements are entered and the file saved. In case, there are no "corrective actions," taken as the editor does not know this is supposed to be a "source file" as opposed to notes for class. If working in a position-dependent language like Java, the developer would have to be very careful about indenting. The file has to be saved with the correct file extension and in a directory where the compiler can find it. Each source file has to be compiled separately, if the program has a few source files, they all have to be named separately in the compiler. When invoking the compiler, it has to be directed to look in the correct directory for the source files and where the output files should be stored. If there is an error in the source file, the compiler will output messages and fail to complete. For any errors, the developer goes back and edits the source file, working from line numbers and compiler messages to fix the problems and these steps continue until all the source files compile without errors. When linking, each object file is specified as being part of the build. Again, the locations for the object files and executable are given. There may be errors at this point because it is not until the entire program is linked that some errors can be detected. Assuming the linker finds all the variables and functions, it produces a file that can be run. If the program is running and working means all well for the developer. If it seems to do nothing, that means it's debugging time, since there is no insight to what the program is doing, the developer may go back and put in some brute force methods, like print statements to print messages out at certain points in the program or blink some light emitting diodes (LEDs) at strategic places, which means back to the editor, and the cycle continues. These are the causes of the developer can sort of sounds like rather than copying and pasting code from previous applications into a new one, that are rewriting the same functionality again and again. This is one of the varied reasons of the dry principle is geared more towards not having the same functionality duplicated throughout an entire system, but reusing code from other applications is better than rewriting it, since IDE is free and available as open source in any codes of environment. The main focus of the study is the development of utility for locking editor in protecting software exercises. Specifically, it sought to solve the following: To develop an editor that will lock the source code during programming laboratory exercises; To assess the propose system in Kirkpatrick model approach; To evaluate the testing assessment of the proposed system in terms of the following indicators: Efficiency, Usability, and reliability As the result, the locking editor for source code generation is a core program for the locking IDE software. It contains both the user interface such as students and instructor which work together in one package. The evaluation of multi-programming environment by testing the editor that, as it develops solutions keep in mind that locking editor uses the architecture and frameworks that most users are accustomed to see. It should build the development environment and use the tools in the way that makes the most sense for organization. The assessment of the proposed system in Kirkpatrick model approach such as reaction, learning, behavior and results with a weighted mean of 4.19, 4.37, 4.25 and 4.67 with a verbal interpretation of "Protectable" and "Highly Protectable" respectively. The determination in testing assessment of the proposed system based on ISO 9126 reveals of efficiency, usability and reliability with a weighted mean of 4.19, 4.58, and 4.75 interpreted "Highly Protectable" and "Protectable" respectively were the main features and characteristics of locking editor.

Index Terms: kirkpatrick, protectable, micro services, programming.

1 INTRODUCTION

The developing executable software in any environment needs to create source files, compile the source files to produce machine code or object files, and link the object files with each other and any libraries or other resources required to produce an executable file. Since, source files contain the code statements to do the tasks the program being created for in which contain program statements specific to the language using [1]. Rather than performing all the steps required to make an executable program as unrelated individual tasks, it brings all the tools needed into one application and workspace. Each of the tools has an awareness of the environment, and they work together to present a seamless development set for the developer called Integrated Development Environment (IDE). The IDE is a software application that provides a programming environment to streamline developing and debugging software. Generally, it provides an easy-to-use interface, automates development steps, and allows developers specifically the students in the academe at the computer laboratory to run and debug programs all from one screen. It can also provide the link from a development operating system to an application target platform, like a desktop environment, smartphone or microprocessor. In other words, it has easy learning programs on the target system, to compile code and to collect result which easy to access codes and this means of open source [2]. The computer laboratory of AMA University is in the set-up of free software which every time and then can be used since have not been authenticated by the Information technology data Center. It means that this open source has referring the students can free to modify and share because its design is publicly accessible to anybody. In other words, the context of software development designates a broader set of values called "open source way" which mean can copy the source code without permission to the owner. The intention of the researcher has to develop a design as a tool for locking the IDE to avoid hacking codes and to make a necessary of ethical hacking, hence the study.

1.1 Background of the Study

The student of AMA University persistence in computing which has the keys to providing their talent needed to fill the computer laboratory in the computing professions. A range of factors can affect a student's decision to remain in a computing major or change to another major if ever they feel that computing education is difficult. This has to describe the

activities in computer laboratory specifically exercises, machine problems, and computing case studies interacting different application programs as the basis of their skills and knowledge in programming capability. The nature of those activities addresses by using of IDE as open source in all programming applications which may result of specific intervention such as using the editor to create a source file, the code blocks, comments, and program statements are entered and the file saved. In case, there are no "corrective actions," taken as the editor does not know this is supposed to be a "source file" as opposed to notes for class. If working in a position-dependent language like Java, the developer would have to be very careful about indenting. The file has to be saved with the correct file extension and in a directory where the compiler can find it. Each source file has to be compiled separately, if the program has a few source files, they all have to be named separately in the compiler. When invoking the compiler, it has to be directed to look in the correct directory for the source files and where the output files should be stored. If there is an error in the source file, the compiler will output messages and fail to complete. For any errors, the developer goes back and edits the source file, working from line numbers and compiler messages to fix the problems and these steps continue until all the source files compile without errors. When linking, each object file is specified as being part of the build. Again, the locations for the object files and executable are given. There may be errors at this point because it is not until the entire program is linked that some errors can be detected. Assuming the linker finds all the variables and functions, it produces a file that can be run. If the program is running and working means all well for the developer. If it seems to do nothing, that means it's debugging time, since there is no insight to what the program is doing, the developer may go back and put in some brute force methods, like print statements to print messages out at certain points in the program or blink some light emitting diodes (LEDs) at strategic places, which means back to the editor, and the cycle continues. These are the causes of the developer can sort of sounds like rather than copying and pasting code from previous applications into a new one, that are rewriting the same functionality again and again. This is one of the varied reasons of the dry principle is geared more towards not having the same functionality duplicated throughout an entire system, but reusing code from other applications is better than rewriting it, since IDE is free and available as open source in any codes of environment. In view, the main focus of this project is to lock the IDE and to develop a new one which has the standard in an ethical hacking for the students' retention knowledge and programming experience with their own capacity.

2 REVIEW OF RELATED WORKS

A Programming language is used to create computer programs such as applications, utilities and system programs. Programming Language is a vital element in today's Software Development [3]. Integrated development environments are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces. IDEs present a single program in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying and debugging software. This study aims to develop a development environment that can be used for Java, VB, C, C++ and C#. The said programming languages are widely used in academia. Programming is a tough course in pursuing any computing degree, student find difficulty studying it. One of the reasons why the student fails in programming is they can't create their own program because they could just easily copy or load the code from other sources. The researchers came up with the development environment for Java, VB, C, C++ and C# which the student can't copy or load code from the computer. The proposed development environment will lock the computer and there will be no other application that can be used. Prior to the existence of high level programming languages, computers were programmed one instruction at a time using binary or hexadecimal notation. This was a tedious job and there were lots of errors. Programs were difficult to read, and modification was extremely complicated because all programs have to be written using absolute addressing. Obviously, this kind of job did not attract many people, resulting in shortage of programmers. Expensive computers sat idle for long periods of time while the software was being developed. Software often cost two to four times as much as the computer. This led to the development of assemblers and assembly languages. Programming became somewhat easier, but many users still wanted floating point numbers and array indexing. Since these capabilities were not supported in hardware, high level languages had to be developed to support them. C# and VB.Net is an example of high level programming language and considered as the most widely used programming language in academia and industry. IDEs initially became possible when developing via a console or terminal. Early systems could not support one, since programs were prepared using flowcharts, entering programs with punched cards (or paper tape, etc.) before submitting them to a compiler. Dartmouth BASIC was the first language to be created with an IDE. Its IDE was command-based, and therefore did not look much like the menu-driven, graphical IDEs prevalent today. However, it integrated editing, file management, compilation, debugging and execution in a manner consistent with a modern IDE. Maestro I is a product from Softlab Munich and was the world's first integrated development environment 1975 for software. Maestro I was installed for 22,000 programmers worldwide. Until 1989, 6,000 installations existed in the Federal Republic of Germany. Maestro I was arguably the world leader in this field during the 1970s and 1980s. Today one of the last Maestro I can be found in the Museum of Information Technology at Arlington. One of the first IDEs with a plug-in concept was Softbench. In 1995 Computerwoche commented that the use of an IDE was not well received by developers since it would fence in their creativity. As of March 2015, the most popular IDE's are Eclipse and Visual Studio. This is intended to clarify and understand better the study through different ideas from previous studies and writing, which are significant to the problem under investigation. The proposed software is application software which can help students to enhance their programming skills in C, C++, C#, Java and VB. Programming language is one of the difficult subjects in IT or computing related courses. Programming language for Computer Science is also a difficult course for freshmen and sophomores because it is a hybrid course [4]. The content is mathematics on many of its applications. Programming languages really challenge computer science students. Many students found an introduction to programming language course more challenging than other courses they have previously taken. One reason, according to Baugh (2005)

is that one of the primary goals of this course is to teach logical reasoning and problem solving. Man cannot learn to program just by reading a book. It is a skill that can be developed through practice. However, Preiss (2006) said that, wise practitioners study the works of others and incorporate their observations into their own practice. Learning your first programming language is probably the most challenging. Consequently, according to Rosen (2005) once you get one under your belt, you'll find that the concepts from language to language have plenty of similarities, so additional languages become easier to pick up. In order to accomplish and implement this proposed software the researcher will use Microsoft Visual C# 2010. This programming language was used for the development and functionality of the software. C# may be the most-used programming language in the world because it makes Windows programming so easy. One of C# strongest selling points is its featured ability to put together a simple program quickly – as well as write very sophisticated applications Computers changed the way how people do their work since it really make their work easier. This benefit provided by computers, lots of people welcome the advancement of technology. Computerization means simplifying production speed and accuracy on the work elements associated with mass production It is the most reliable source of information in any field of studies nowadays so students really need computers to uplift their study. Educators agree that technology plays crucial role in transforming classroom environments to enhance teaching and learning. The accomplishment of the software would greatly help the students in their studies. A compiler is a computer program that transforms source code written in a programming language (the source language) into another computer language (Esparrago, et. al. 2006). Computers are so powerful and useful for the convenience and comfort of the human life (Pepito, 2005). The development of the proposed software will surely benefit the students and instructors. The researcher proposed this study because he wants the students to have an easy way of understanding and learning. Computers have changed the face of education all over the world (Velasco, 2005). Evaluation in terms of training effectiveness is beneficial to both students and instructor. Therefore evaluation has been conducted as the last step of training cycle. Its main objective is to articulate impediments at individual or in the laboratory classroom. Kirkpatrick & Kirkpatrick (2006) pointed the reason why measurement of training effectiveness is required more in details; (1) to judge continue or scrap the program, (2) to judge its relevance to the objectives, (3) to know-how to improve it, (4) to justify its budget, and (5) to prove its necessity.

Fig. 2.1 Kirkpatrick's Four Level Model

Figure 2.1 shows the Kirkpatrick's four-level model that each successive evaluation level is built on information provided by the lower level.

Assessing training effectiveness often entails using the four-level model developed. According to this model, evaluation should always begin with level one, and then, as time and budget allows, should move sequentially through levels two, three, and four. Information from each prior level serves as a base for the next level's evaluation. Thus, each successive level represents a more precise measure of the effectiveness of the training program, but at the same time requires a more rigorous and time-consuming analysis. Training effectiveness refers to the extent to which the training objectives or training's goal are achieved. Most of the research on a training evaluation has been relied the four-level typology to explain the effectiveness of training. Level 1, reaction, is trainees' feelings about and like of a training program, hence evaluated below:

Table 1.0 Dimensions of Reactionnaires

Dimensions	Purpose
Program objectives/content	To evaluate the objectives of a program with participants' expectations.
Program materials	To ascertain the effectiveness, efficiency and usefulness of written and other materials used in the program.
Delivery methods/technologies	To judge the appropriateness and effectiveness of instructional delivery
Instructor/facilitator	To rate the presentation skills, leadership and overall effectiveness of
Instructional activities	To evaluate the appropriateness and usefulness of in- and out- of-class
Program time/length	To assess the time length and sessions of an entire program.
Training environment	To evaluate the adequacy of the physical training environment, including
Planned action/expectation	To evaluate the participants' plans and expectations for applying what was
Planned action/expectation	To evaluate the participants' plans and expectations for applying what was
Logistics/administration	To evaluate the smoothness and effectiveness of scheduling, registration
Overall evaluation	To determine overall participant satisfaction with and feeling about the
Recommendations for program improvement	To solicit suggestions and recommendations for improving this or

Level 2 is learning that has taken place during training. Alliger and Janak, (1989) define learning as the "principles, facts, and techniques understood and absorbed by the trainees" .No change in behavior can be expected unless one or more of these learning objectives has been accomplished (Kirkpatrick, 1994). This level of evaluation allows trainees to demonstrate their understanding of specific within the learning program. Level 3, behavior change or transfer, refers to the extent to which change in behavior has occurred because the trainees attended the program, which is measured of assessment in the workplace. This level attempts to determine whether trainees who can apply the acquired specific knowledge and/or skills, use their new knowledge and/or skills when returning to the work environment. In the IEEE's evaluation of behavior based on the Software Engineering Body of Knowledge (SWEBOK), Carrington (2014) specified the goals that a software development environment includes IDEs should meet. Such environments should (1) reduce the cognitive load on the developer; (2) free the developer to concentrate on the creative aspects of the process; (3) reduce any administrative load associated with applying a programming method manually; and (4) make the development process more systematic. This may reveal a

usability of software in process of overall evaluation of how a system performs in supporting a user/student for a particular task. Level 4, results, refers to the final results that occurred because the trainees attended the program. This level four evaluation attempts to assess training in terms of proposed system results. In this case, the software program improved steadily training for the students. Frequently thought of as the bottom line, this level measures the success of the program in terms that students can understand by improving their knowledge, their logical thinking about analyzing the instruction, and widened their algorithm development logic. Micro services (MS) is a software architecture style in which a large complex software application is decomposed into many services each of small hence micro size (Lewis et. al., 2015). These services can be independently deployed to a cluster of computers and duplicated to achieve load balance and system-performance optimization. Each runs in its own process and interacts with other services through lightweight communication mechanisms. The alternative is the so-called monolithic style, in which number of services is small. MS brings many benefits for engineering service-oriented systems. These include support for team development, continuous testing and seamless integration. It is widely considered to be an effective architecture for cloud computing and service-oriented applications (Scalability, 2015). However, developing a system that consists of a large number of micro-scale services running on a farm of servers imposes grave challenges to software engineering (Newman & Krause, 2015). These include: Program complexity, due to the thousands of services at micro scale running asynchronously in a distributed computer network. Programs that are difficult to understand are also hard to write and hard to modify; Performance criticality, especially because MS is often adopted in order to improve performance, and this requires the system to be elastic according to the workload. MS enables services to be duplicated and distributed to a cluster of servers but the choice of where to duplicate and distribute to can have a great impact on performance. We must therefore be able to test systems with different duplication and distribution patterns; Evolution continuity, which is often required by the application domain for which MS has been chosen. MS enables flexible modification of the component micro services and seamless integration of new functionality into an existing system. However, ensuring this advantage is a difficult task, because the development and testing must be done in a cluster environment. The solution to the complexity challenge is a novel programming language called CAOPLE (Xu et. Al, 2016) where demonstrated that it can be used to program service-oriented systems as part of MS. COAPLE is based on the caste-centric conceptual model of multi-agent systems (Zhu, 2013). It has been implemented by compiling high-level source code into object code for a lightweight language-specific virtual machine called CAVM-2, which is a complete redesign of an initial prototype called CAVM. In the study of integrated software development environment (Liu et. al, 2016) called CIDE (CAOPLE Integrated Development Environment) that supports programming in CAOPLE. Most importantly, it supports the precise control over deployment and testing that is necessary to overcome the performance criticality and evolution continuity challenges mentioned above. For more information, it is to determine the following key features of CAOPLE Language which stands for Caste-centric Agent-Oriented Programming Language and Environment (Zhu, 2013). These agents are service providers that are analogous to real-world counterparts such as estate agents that buy and sell properties, and travel agents that buy and sell air tickets. It is worth noting that agents can themselves be service requesters. In the literature on service-oriented architectures, including that on MS, the word “service” can mean either the functionality provided by the computer system (Singh & Huhns, 2015) or the computational entity that provides that functionality. Here, we use the word “service” with only the first meaning, reserving the word “agent” for the second. In our conceptual model, an agent is an autonomous entity running in its own process. Agents encapsulate data, operations and behaviour rules. They execute in parallel and cooperate with each other through a set of well-defined asynchronous communication channels. This corresponds exactly with the second meaning of services. CAOPLE also introduces the classifier of agents as a language facility, which is called a *caste.* Therefore, agents are runtime instances of castes just like objects are runtime instances of classes. Conversely, a caste is a template for agents just like a class is a template for objects. The word “microservice” can either mean one of several identical copies of a service, each being an agent in our conceptual model, or it can mean a template from which instances can be generated and deployed to different servers, i.e. a caste in our conceptual model. So the concepts of caste and agent that we have just introduced correspond precisely to these two different meanings. However, these are implementation of COAPLE on CAVM Virtual Machine COAPLE is implemented as the compilation of source code into object code of the CAVM-2 virtual machine, which is designed for languages like CAOPLE running in a distributed computing environment. CAVM-2 provides a runtime environment in the same way that Java Virtual Machine (JVM) supports the execution of Java program. On the other hand, CAVM-2 is significantly different from the JVM in both architecture and functionality. CAVM-2 consists of two key elements: Communication Engine (CE) provides a lightweight communication mechanism for network transparent communications between agents; Local execution engine (LEE) executes the instructions of the program code and fulfills the computation logic. These elements can be distributed on each hardware node of a cluster in any combination of CE only or LEE only or both CE and LEE. This is illustrated in Figure 1. Agents can be executed on any LEE and communicate with other agents through the CE without knowing their physical location on the network, so CAOPLE code can be network transparent. More details of the virtual machine and its implementation can be found in.

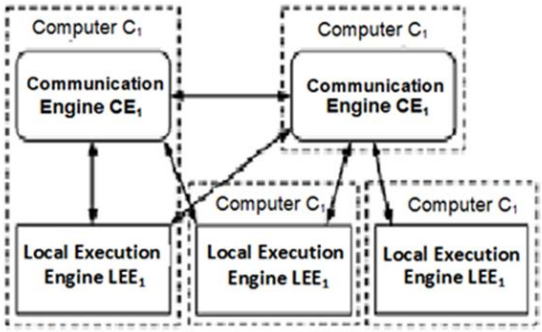

Fig. 2.2 Overall Structureof CAVM Virtual Machine

In achieving optimal performance for the cluster, it is important to allocate carefully the CEs and LEEs to the computer nodes, to deploy and distribute the castes to the CEs and to allocate and distribute agents to various LEEs in the network. As the workload profile of an application changes dynamically, these CE/LEE allocations and caste/agent distributions must be adapted flexibly. The CAVM-2 virtual machine provides a flexible mechanism for such configuration of the cluster and deployment and distribution of agents. However, the challenge that remains is how to turn such a mechanism into a facility that enables the developers to test and debug such a system in various platform configurations and code distributions. Application Programming Interfaces (APIs) provide ready-to-use functionalities to alleviate the complexity of coping with common development tasks. Understanding how to properly use APIs of different libraries and frameworks is a non-trivial and time consuming task. For instance, developers should invoke methods in specific orders and follow the control structure constraints among the method calls. Otherwise, the normal execution of the program may become disrupted and the program raises an exception. Relying on the APIs, the query formulation (Amintabar et. al., 2015) that when an exception occurs, ExceptionTracer uses the stack trace information to locate the API which caused the exception. It then constructs the Groum representation (Nguyen et. al., 2011) of the code, i.e., a directed graph that represents how one or multiple objects are used in the code. This graph includes orders of method calls, as well as control and data dependencies among them. ExceptionTracer explores this graph to identify other objects that are involved with the API which caused the exception. The information about these objects that includes the type of each object and the invoked methods, together with the API which caused the exception constitute the contextual information likely relevant to solve the problem. For instance, Figure 4 demonstrates the Groum of our exemplary code snippet. We first mark the node labeled FileInputStream.read that corresponds to the statement which caused the exception in the source code. We name this node ExNode. With respect to the binding information in the AST representation of the source code, we afterwards mark any adjacent node to the ExNode, if they are in the same library with the ExNode. Similarly, it mark any adjacent nodes to a marked node if they belong to the same library. This process leads to mark all the program statements of the same library involved with the ExNode and reduces the Groum

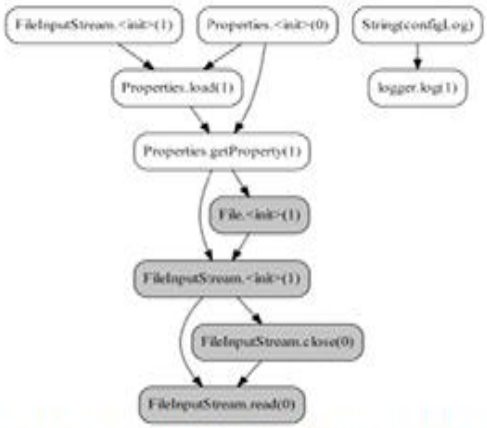

Fig. 2.3 The Interplay Between Different Objects in Exemplary Code Snippet

graph to the one shown in Figure 5. This graph represents the contextual information of the exception. The plugin then formulates two queries, the first one is a BOA query (Dyer et. al., 2013) which enables us to mine Java projects in Source-Forge and and second is a textual query that combines the exception message with the contextual information extracted earlier to search discussions in Stack Overflow. Technology has a great impact on our society nowadays. Education was totally improved by the application of computer technology. It was at this point where the researchers arrived with the idea of creating software to help individuals who are willing to learn and acquire information. Economically investments in computer and internet access have powerful benefits in enhancing the educational system of the country. Many researches proved that employing technology for teaching and learning yield positive results for students and teachers alike. With sufficient access and support, teachers were able to help their students comprehend difficult to understand concepts and engage them in learning, effectively, provide their students with access to information and resources, and meet their student's individual needs better. (Baldonado et.al, 2005) The modern technology we have now, like computers have a tremendous role in helping people make their work easier and faster. By using a computer, the researchers were able to find a way where users can learn the importance of its existence especially for the students. The advent of computers has made it possible for people to improve information-related processes that are normally time-consuming and require a lot of mental dexterity. Work that normally has to be done by experts can now be done by non-experts because of computers (Coronel, 2005). Learning with the use of computers makes learning easier and it lessens the time consumed in studying. Learning and acquiring information and knowledge are very time-consuming, but with the help of computers, everything becomes efficient and convenient (Aldoz et.al, 2005). Computer is used to assist in the educational process itself. Computers are considered to have contributed a lot in our society including educational system for it has proved to be a valuable educational tool (Julian et.al, 2003). Computer application programs contribute great significance in accomplishing tasks without increasing the need of manpower. Difficulties encountered in completing a certain task had always been an issue in man's race towards work advancement. Man always find ways to accomplish task in the quickest, accurate and efficient manner. Hence, computer technology has truly a big part in our everyday life (Delos Santos and Nugera, 2005). Some schools teach limited programming language, so students are compelled to learn new programming language on their own. The problem now is how computers can help students to learn new programming languages in more convenient and easier way (Marbella and Sepina 2007). Thorough reading of some unpublished materials, the researchers found many important insights. Included are the writings and research of unpublished thesis of students, which would serve as a guide in the purpose of having a solution for the study. Programming languages exist only for the purpose of bridging the gap in the level of abstraction between the hardware and the real world. There is an inevitable tension between higher levels of abstraction that are easier to understand and safer to use, and lower levels of abstraction that are more flexible and can often be implemented more efficiently. To design or choose a programming language is to select an appropriate level of

abstraction, and it is not surprising that different programmers prefer different levels, or that one language may be appropriate for one project and not for another. Within a special language, a programmer should understand in depth the safety and efficiency implications of each construct in the language (Rohe, 2004). The researchers proposed the study not just only for the benefit of the students, but also for the teachers themselves for they can use it as an assessment tool in teaching, especially those teaching programming language. The purpose of this thesis is to introduce programming language and help educate others on topics that can be implemented in the curriculum (Boyd, 2003). Many programmers have poor linguistic skills, they are passionately in love with their 'native' programming language, thereby neglecting to analyze and compare language constructs, or to understand the advantages and disadvantages of modern languages and language concepts (Zager, 2009) Programming myopia occurred when programmers become so fascinated with the interesting special properties and optimization opportunities of a data structure or algorithm they have discovered, thus, they fail to recognize its sub optimality in general (Boyd, 2009). The study aimed to recommend Graphical User Interface feature which is the best suited representation. Through this study, Novion cited the GUI allows users to visualize the software in a friendly manner. The GUI uses objects to represent the software tools and make clearer in presenting the software (Novion 2009).

2.1 Synthesis

The related literature and studies discussed by the researchers contributed in conceptualizing the proposed system should be composed of, should have features and the possible benefits that could offer to the beneficiaries. Related literature and studies had shown the relevance of the proposed system in academia. Computers are considered to have contributed a lot in our society, including educational system for it has proved to be a valuable educational tool. The proposed study would help the students to improve their programming skills in C, C++, C#, Java and VB. Programming language is one of the difficult subjects in IT related courses. There should be a tool or development environment that helps the student in studying programming. The researchers proposed this study not just only for the benefit of the students, but also for the instructors themselves since they can use it as an assessment tool in teaching programming language.

2.2 Conceptual Framework

The researcher will define an integrated development environment (IDE) or interactive development environment that provides comprehensive facilities to computer programmers for software development. It normally consists of a source code editor, build automation tools and a debugger. This also contains a compiler, interpreter, or both, such as NetBeans and Eclipse. The proposed software is a Locked Development Environment for C, C++, C#, Java and VB that can edit source code and compile.

Fig. 6 The Conceptual Paradigm of Lock Development Environment

The figure above shows the systems works with the developers in which the instructor can assess his student. This is the session at the laboratory class where the instructor can give specific time and kind of application to be used. The student will use the session file to create a program based on the specified settings of the session. Once the software has turned-on the computer will be locked automatically and there will be no other application can be used and will run with specific given time. The student can only compile and run their program codes based on the specified programming language.

3 OBJECTIVES

The main focus of the study is the development of utility for locking editor in protecting software exercises. Specifically, it sought to solve the following:

1. To develop an editor that will lock the source code during programming laboratory exercises.
2. To assess the propose system in Kirkpatrick model approach.
3. To evaluate the testing assessment of the proposed system in terms of the following indicators: Efficiency, Usability, and reliability

4 METHODOLOGIES

This section presents the methods and procedure needed to develop the proposed software. It also discusses the research design, data gathering instruments, analytical tools as well as statistical tools.

4.1 Research Design

The researcher will produce system design appropriate to the propose system which technique must in order to attain the objectives and its goals. It will also to use descriptive statistics to get the frequency, percentage and mean on the evaluation of software testing based on respondents' assessment.

4.2 Methods of Research Used

The researcher will use a descriptive method of research which has a process of data gathering as well as analyzing, classifying, and tabulating data. It also a basis of this method, the information gathered in the existing system to be the useful for designing to strengthen the weakness.

4.3 Respondents of the Study and Sampling Techniques

The project will evaluate by thirty (30) respondents specifically the students in random sampling techniques. The respondents will be proficient in C. C++, C# or in visual basic programming from College of Computer Studies. Each evaluator will be given a set of questionnaires for them to fill up as the basis for their evaluation. The results will be tabulated and computed

using the Mean Range Formula to conclude if the system software meets the Software Quality Factors acceptance. A set of evaluation questionnaires will be used for the final evaluation of the system software to be developed.

4.4 Data Gathering Instruments

This has the researchers' basis to collect additional information for the accomplishment of the study which essentially to acquire a reliable information needed for implementation.

Questionnaire. This type of data gathering instrument where in the respondent will evaluate the proposed system which the criteria given to be assessed on database security. It employs the Likert Type, with five alternative responses assigned equivalent points are as follows.

Verbal Interpretation	Weight	Interval
Highly Protectable (HP)	5	4.51 - 5.00
Protectable (P)	4	3.51 - 4.50
Moderately Protectable (MP)	3	2.51 – 3.50
Slightly Protectable (SP)	2	1.51 – 2.50
Not Protectable (NP)	1	1.00 – 1.50

with a use of weighted mean that the central tendency of raw data and used present the assessment of the respondent on the proposed system.

4.5 Software Development Methodology

The researcher will use rapid application development (RAD) model as software development methodology that splitting a software development work into distinct phases or stages. This model distributes the analysis, design, build, and test phases into a series of short, iterative development cycles. These are the following phases:

Model-Business Modeling. This has the researchers' propose system under development is designed in terms of flow of information and the distribution of information between various processes. A complete activities analysis is performed to find the vital information on how it can be obtained, how and when the information will be processed and what are the factors driving successful flow of information.

Data Modeling. The information gathered in the first phase will be reviewed and analyzed to form sets of data objects vital for the propose system. The attributes of all data sets will identify and define. The relations between those data objects are establish and define in detail in relevance to the business model.

Process Modeling. The data object set will be defined in the Data Modeling phase are converted to establish the business information flow needed to achieve specific business objectives as per the business model. The process model for any changes or enhancements to the data object sets is defined as phase-process descriptions. It is for adding, deleting, retrieving or modifying a data object are given.

Application Generation. The actual system is built and using automation tools to convert process and data models into actual prototypes does coding. Testing and Turnover the overall testing time is reduced in RAD model as the prototypes are independently tested during iteration. However the data flow and the interfaces between all the components need to be thoroughly tested with complete test coverage. Since most of the programming components have already been tested, it reduces the risk of any major issues. The following functionalities need to be defined will start with identifying the procedures to be made in the proposed inventory system via Standard Operating Procedures (SOP's). Next to it must train the students in the use of the propose system, and build their own capacity of programming. It also needs to protect the system from technical faults and from fraudulent use by ethical hacking. This will follow by examining thoroughly the propose system for data accuracy and efficiency. Lastly, determine the Kirkpatrick's four level model by assessing through the use of the propose system in a trial run in terms of technical and operational parameters. Once those functionalities will determine, it can follow a model as depicted shown in figure 2. This method starts with objectives of the study, with a test at each step and a repeat iteration if the whole system is still inefficient.

Figure 8.0 Rapid Application Model

The figure above shows the Rapid Application Development. This is a software development methodology uses minimal planning in favor of rapid prototyping. A prototype is a working model that is efficiency equivalent to a component of the propose system and these are the following:

Define. The researcher used unstructured interview technique to obtain the information from the IT data center personnel about the content and features undertakings for the students, the findings were used to be the reference for the proposed system.

Analyze. The researcher analyzed and identified all requirements needed for developing the design by combining all suggestions from the target. The results were converted to a template, which the locking editor used to generate the integrated development environment.

Design. This phase describes the desired features and operations in detail, including screen layouts, user-interface design and application design. The figure below shows the

Steps followed in the system design phase:

Figure 10.0 System Design Phase

Layout design. It is related to the cognitive arrangement of different programming languages that convey the viewer's adequately.

User-interface design.This is critical to the program in terms of the suitability and usability for the users. The design and evaluation process involves user interaction at every exercise.

Application design. Determines how the finished program works. IT describes the flow of the functions and data structure used in the proposed system

Develop. The actual code/program was written using Java programming language to develop the entire modules. In this phase, the researcher encountered many problems when tried the software design to the actual Java code. There was a difficulty in converting the layout design and use interface design to Java code because of the IDE via Internet does not have visual view, thus the researcher could not see how the result looked like until compilation.

Test. The researcher used software testing environment to check for errors, bugs, and inter-operability using general test procedure for unit testing such that individual components were tested to ensure that operated correctly and each component was tested independently, without other components.

Support. The researcher used references such as books and other materials for programming exercises that support the system.

4.6 Analytical Tools

The researcher will use the following tools and techniques for strategizing the propose system.

System Flowcharts. It is to present a comprehensive picture of the management, operations, information systems, and process controls embodied in the operation of the propose system.

Data Flow diagrams (DFD). This is to portray process activities, stores of data, and flows of data among those elements.

Use-Case Diagram. It is a set of use cases or a model of the interaction between external users of a software product (actors) playing a specific role, describing a set of user scenarios, capturing user requirements, and contract between end user and software developer.

5 PRESENTATION, INTERPRETATION AND ANALYSIS OF DATA

5.1 Locking Editor Source Code Generation

Generating locking editor source code is the main package of software exercises for the students at the computer laboratory. It is the front-end where the students can interact with the program and also the center-point for the activities depends on the usage programming language interests. This locking editor generation source code has consists of menu to operate together that the main objective of this package is to acquire programming ability or data programming from the instructors' instruction and make use them to generate code for the courses. The below figure shown how the locking editor are related to each other.

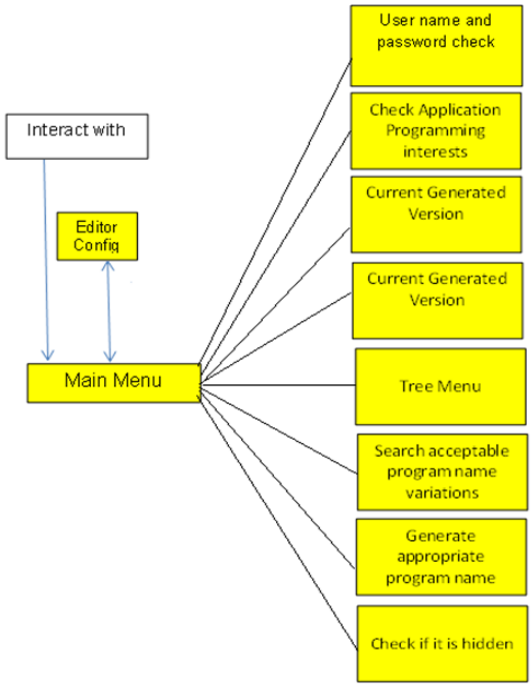

Figure 13.0 Locking Editor Source Code Generation

On the figure, the one way arrow represents the relation interaction of the students to the editor, where the arrow pointed to the config editor via the main menu. The user name with its password interfacing of the students who usually doing the programming exercises, where it has also checking of the application align to their interests of the courses, that also to present the current version for updating the release of the software. Tree menu has ordered through the operation of the editor to the instructors' instruction to give necessary activities relying to IDE. On the other hand, search program name variations where there are selections of the students as a user have their own file for their activities in their respective courses, where a generate appropriate program name that can be used any programming language import or export only source code of the specific students as is own files that must also to check if it is hidden or not.

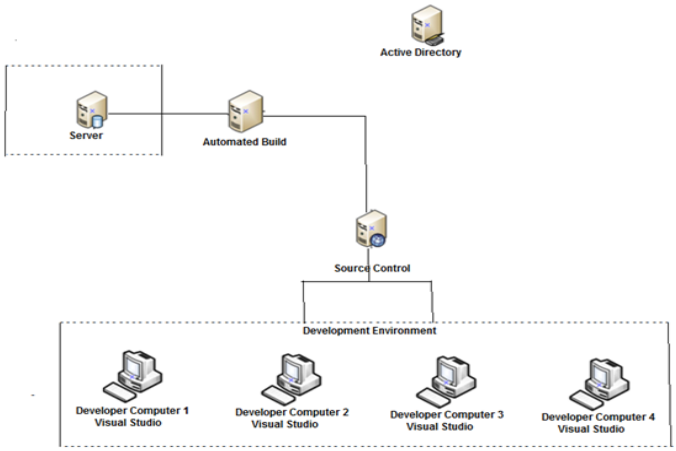

Figure 13.0 Multi-programming Environment by Testing the Editor

The development environment is obviously uses a development environment to build and test any programming changes before updating the production system. When setting up different software programs development environment, it must consider the following factors such as the number of discrete projects or software programs required, the number of users, the types of customizations required, deployment configurations, available hardware, and iteration frequency and deployment schedule. The server environment development requires several infrastructure components such as Active Directory and Server. However, it is not convenient to set up different physical server environments for developing and testing. This approach has the following advantages such as complete isolation of the development experience, allowing for the developer to maintain his or her environment as appropriate, simplified debugging of individual code, and easier develop custom Visual Studio since it can reference and debug the pages directly on the server.

Assess the Propose System in Kirkpatrick Model Approach

Table 4.0

Assessment of the Respondents in terms of Reaction

	Weighted Mean	Verbal Interpretation
1. The instructors' contribution to the training process was valuable	4.20	P
2. The materials used as well as hand-outs are making useful for the topic exercises.	4.11	P
3.The method of teaching is anchored to exercises aligned an matched.	4.26	P
Total Average Mean	**4.19**	**P**

Table 4.0 presents the assessment of the respondents in terms of reaction. All items are "P" in verbal interpretation such as "The method of teaching is anchored to exercises aligned and matched", "The instructors' contribution to the training process was valuable", and "The materials used as well as hand-outs are making useful for the topic exercises" with a weighted mean of 4.26, 4.20, and 4.11 respectively. It gleans from the table in terms of reaction that the proposed system enhancement is a training evaluation that can be described as a systematic process of collecting and analyzing information for and about a training program which can be used for planning and guiding decision making as well as assessing the relevance, effectiveness, and the impact of various training components.

Table 5.0

Assessment of the Respondents in terms of Learning

	Weighted Mean	Verbal Interpretation
1.The instructors have responsibility o reporting student learning accurately and fairly, based on evidence obtained from a variety of contexts and applications.	4.45	P
2. Students understand exactly what they are to learn, what is expected of them and are given feedback and advice on how to improve their works.	4.13	P
3.The environment of editor provides complete feature that makes user-friendly approach.	4.55	HP
Total Average Mean	**4.37**	**P**

Table 5.0 presents the assessment of the respondents in terms of learning. The "The environment of editor provides complete feature that makes user-friendly approach", leading, with a mean of 4.55 and verbal interpretation of "HP" followed by "The instructors have responsibility of reporting student learning accurately and fairly, based on evidence obtained from a variety of contexts and applications", with weighted mean of 4.45 and verbal interpretations of "P", and lastly the "Students understand exactly what they are to learn, what is expected of them and are given feedback and advice on how to improve their works" with only mean of 4.13 and has a verbal interpretation of "P." Gleaned from the table that learning which are keen to improve their productivity, efficiency and profitability will look to move beyond training and look at more diverse learning and development activities which will enable the students to maximize their potential and provide a valuable resource for the laboratory classroom.

Table 6.0

Assessment of the Respondents in terms of Behavior

	Weighted Mean	Verbal Interpretation
1.Navigation tools provided by the software such as menu or tree views are appropriate for the software.	4.40	P
2. The system uses confirmatio to assure execution of specified task such as when the user wants to close a project without saving it.	3.91	P
3. The system provides complete instructions to advise the user what to do when problem occurs.	4.45	P
Total Average Mean	**4.25**	**P**

Table 6.0 presents the assessment of the respondents in terms of behavior. Majority of the respondents gives all items in a verbal interpretation of "P" with a weighted mean of 4.45, 4.40, and 3.91 respectively such as "The system provides complete instructions to advise the user what to do when problem occurs" which leads to all items, it follows the "The system provides complete instructions to advise the user what to do when problem occurs", and lastly the "The system uses confirmation to assure execution of specified task such as when the user wants to close a project without saving it." The table reveals in terms of behavior that there must be a provision for development opportunities alone with the students do mean that will be more productive and effective software exercise through teaching and learning activities. This has the opportunity need to be appropriate in terms of content and the way that it is delivered so that they will add more knowledge and skills to the overall computer classes. Training programmes are effective only to the extent that the skills and behaviors learned and practiced during instruction are actually transferred to the right computer laboratory workplace.

Table 7.0
Assessment of the Respondents in terms of Results

	Weighted Mean	Verbal Interpretation
1.The proposed system seemingly reasonable to maintain as a support to the computer facilities.	4.60	HP
2. It is feasible for having the proposed system as a development for the students testing basis to improve their algorithm development logic i programming language.	4.75	HP
3.This has absolutely related training necessary to outfit th computer laboratory testing ground to mean programming language.	4.66	HP
Total Average Mean	**4.67**	**HP**

Table 7.0 presents the assessment of the respondents in terms of results. Take note that all criteria have the same verbal interpretation of "HP." It leads by "It is feasible for having the proposed system as a development for the students testing basis to improve their algorithm development logic in programming language" with weighted mean of 4.75 as the highest, while the "This has absolutely related training necessary to outfit the computer laboratory testing ground to mean programming language" with a mean of 4.66 as second, and "The proposed system seemingly reasonable to maintain as a support to the computer facilities" as the last. The table of assessment in terms of results has more on protection of software exercise in a tool of locking editor. This merely an evaluation of usability of the proposed system and protect ability the knowledge and skills of the

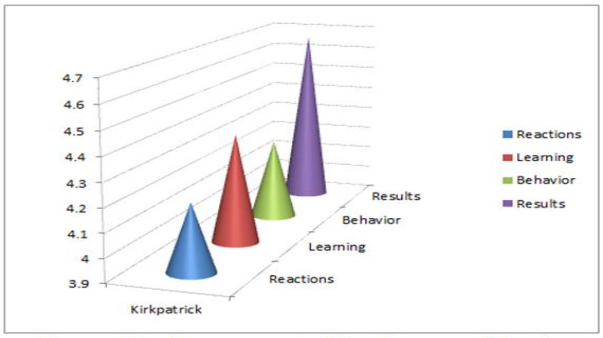

Figure 14 Assessment of the Proposed System in Kirkpatrick Model Approach

students with their capability to trigger the difficult machine problem as such with no basis and intervention leading to hacking source code. Figure 14 presents the assessment of the proposed system in Kirkpatrick approach based on the outcome of respondents' evaluation. The model has the advantage of being consistent in utility protect in software exercises for the students in computer laboratory. The analysis indicates that this model would most accurately reflect the actual ongoing operation in programming language. It has a better proportionate sharing of students' capability in a sense that they have their enhancement with capabilities, thus adequately recognize the IT data center for the following objectives to serve framework: Provide a program of computer studies that supplements the computer education program being standardization of protecting software.; Provide students and other colleges inside AMA University with up-to-date training and practical experiences in skills that are required for careers in high technology and for coping in an increasingly technological society.; Produce some highly-qualified, motivated students specifically the senior high school students who wish to pursue advanced studies of higher education leading to computer-related professional careers.; Prepare elementary and secondary instructors to teach a range of computer-related topics beginning with computer literacy and continuing through more advanced computer courses aside current programming language.; Develop, test, and disseminate a model for expanding computer education through the pooled resources.; Promote and sustain industry/education cooperation [5].

5.2 Longer-Term Objectives:

- Test and refine new approaches to computer-related training and education.
- Serve as a recognized source of technical assistance in developing state-of-the art computer-education courses and programs of study.
- Serve as a laboratory for testing new hardware, software, and educational products preferring for industry.
- Facilitate changes that enable the university to prepare students for an increasingly technological society and to be more responsive especially to industry's need for a technologically skilled work force.

Testing Assessment of the Proposed System

Table 10.0
Testing Assessment in Terms of Efficiency

	Weighted Mean	Verbal Interpretation
1.The activities given though locking editor has automatic save.	3.80	P
2.The operating system is flexibl in any application of software exercises.	4.55	HP
3.The code editor should provide plug-ins to help students.	4.22	P
Total Average Mean	**4.19**	**P**

Table 10.0 presents the testing assessment in terms of efficiency. Majority of the total respondents interprets "The operating system is flexible in any application of software exercises" as "HP" with a mean of 4.55, while "The code editor should provide plug-ins to help students" and "The activities given though locking editor has automatic save" with a weighted mean of 4.22 and 3.80 respectively and with the interpretation of "P." It gleans from the table that efficiency of the proposed system provides programming capabilities through a text editor or graphical user interface (GUI) and integrate with at least one platform without a separate plugin. It also expose a platform's application programming interface (API) and allow for compiling, debugging, version control, platform-specific code suggestions, or code deployment.

Table 11.0
Testing Assessment in Terms of Usability

	Weighted Mean	Verbal Interpretation
1.It has automatically analyzes user assembly and object code files appropriately.	4.80	HP
2.It eliminates the need for many non-standard application such as C or Java qualifiers and compiler options.	4.55	HP
3.It removes unused functions and variables.	4.38	P
Total Average Mean	**4.58**	**HP**

Table 11.0 shows the testing assessment in terms of usability. The items of "It has automatically analyzes user assembly and object code files appropriately" and "It eliminates the need for many non-standard application such as C or Java qualifiers and compiler options" with a mean of 4.80 and 4.55 with equal interpretation of "HP respectively, and item "It removes unused functions and variables" with weighted mean of 4.38 and interprets "P" that assessing of the respondents are very much close in all items. It reveals from the table that usability of the proposed system integrates a variety of tools for applications such as syntax-directed editors, debuggers, and screen layout tools.

Table 12.0
Testing Assessment in Terms of Reliability

	Weighted Mean	Verbal Interpretation
1. Debug virtual memory applications.	4.60	HP
2. The locking editor is reliable software as a service.	4.90	HP
3. It configures the applications to meet specific requirements.	4.75	HP
Total Average Mean	**4.75**	**HP**

Table 12.0 presents the testing assessment in terms of reliability. Take note that all items such as "The locking editor is reliable software as a service", "It configures the applications to meet specific requirements", and "Debug virtual memory applications" have the same interpretations of "HP and with a weighted mean of 4.90, 4.75, and 4.60. The table reveals that average mean of 4.75 of reliability that the proposed locking IDE is more than just a text editor which time-saving utility for performing repetitive and complex tasks. A set of integrated tools assists with the typical development workflow of coding, testing and debugging, profiling, compiling, running, and deploying. The locking editor has an integrated task list which keeps track of unfixed errors and unimplemented features and lets it jump directly to-do item in own respective source code. With the locking IDE, it is easy to manage its own software projects and maintain control of multiple file revisions in an environment subject in avoidance hacking codes.

6 SUMMARY, CONCLUSIONS AND RECOMMENDATION

This section presents the summary of findings, as well as conclusion and recommendations based on the specific objectives.

6.1 Summary of Findings

The summary of findings based on the interpretation where the researcher exposed to the data gathering and enhancement of proposed system. Apparently, majority of the students specifically the CCS, were the target as a trainees with respect to instructors' activity instructions given them to as a main sources of data gathering, to tackle the problem objectives and is follows:

1. The locking editor for source code generation is a core program for the locking IDE software. It contains both the user interface such as students and instructor which work together in one package.
2. The evaluation of multi-programming environment by testing the editor that, as it develops solutions keep in mind that locking editor uses the architecture and frameworks that most users are accustomed to see. It should build the development environment and use the tools in the way that makes the most sense for organization.
3. The assessment of the proposed system in Kirkpatrick model approach such as reaction, learning, behavior and results with a weighted mean of 4.19, 4.37, 4.25 and 4.67 with a verbal interpretation of "Protectable" and "Highly Protectable" respectively.
4. The determination in testing assessment of the proposed system based on ISO 9126 reveals of efficiency, usability and reliability with a weighted mean of 4.19, 4.58, and 4.75 interpreted "Highly Protectable" and "Protectable" respectively were the main features and characteristics of locking editor.

6.2 Conclusions

Based on the summary of findings, the researcher drawn the following conclusions.

1. The locking editor for source code generation is a core program for the locking IDE software having interface, which interact directly with the user to use their interest of application based on the courses and relay to the process of code editor.
2. The evaluation of multi-programming environment by testing the editor that integrated development environments are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces utilizing different programming languages. It presents the different software programs in which all development is done. This program typically provides many features for authoring, modifying, compiling, deploying and debugging software, and software development using related tools, such as C, C#, VB.net, Java and the like for the applications to use based on students' activities.
3. The assessment of the proposed system in Kirkpatrick model approach led by "Result" due to its more on protection of software exercises in a tool of locking editor that merely an usable. The result of Kirkpatrick model are related to efficiency which not only as of ethical hacking but solely the knowledge and skills of the students to practice their logic in terms on to develop their algorithm capability.
4. The determination in testing assessment of the proposed system were reliable With the locking IDE, it is easy to manage its own software projects and maintain control of multiple file revisions in an environment subject in

avoidance hacking codes and interpreted as “Highly Protectable.”

Acknowledgment

The author wish to thank Paul M. Grafilon and Jonathan Fuller,. This work was supported in part of the completion of Capstone Project.

REFERENCES

[1] S. Bilin Balaji, “What is the difference between source file, executable file and object file?” https://www.quora.com/What-is-the-difference-between-source-file-executable-file-and-object-file

[2] M. Rouse, “Integrated Development Environment” http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment

[3] “Programming Language” https://www.computerhope.com/jargon/p/proglang.htm

[4] L. Elrick, “Is Computer Programming Hard? Not if You have these 6 Characteristics” http://www.rasmussen.edu/degrees/technology/blog/is-computer-programming-hard/

[5] E. Forest, “Kirkpatrick Model: Four Levels of Learning Evaluation” https://educationaltechnology.net/kirkpatrick-model-four-levels-learning-evaluation/