EVIDENCE 1: Introduction to Software

For as long as there has been computer hardware, there has also been computer software. But what is software? Software is just instructions written by a **programmer** which tells the computer what to do. Programmers are also known as 'software developers', or just plain 'developers'.

Nothing much is simple about software. Software programs can have millions of lines of code. If one line doesn't work, the whole program could break! Even the process of starting software goes by many different names in English. Perhaps the most correct technical term is '**execute**', as in "the man executed the computer program." Be careful, because the term 'execute' also means (in another context) to put someone to death! Some other common verbs used to start a software program you will hear are 'run', 'launch, and even 'boot' (when the software in question is an operating system).

Software normally has both **features** and **bugs**. Hopefully more of the former than the latter! When software has a bug there are a few things that can happen. The program can **crash** and **terminate** with a confusing message. This is not good. **End users** do not like confusing **error** messages such as:

Site error: the file /home7/businfc6/public_html/blog/wordpress/wp-content/plugins/seo-blog/core.php requires the ionCube PHP Loader ioncube_loader_lin_5.2.so to be installed by the site administrator.

Sometimes when software stops responding you are forced to manually **abort** the program yourself by pressing some strange combination of keys such as ctrl-alt-delete.

Because of poor **usability**, documentation, and strange error messages, programming still seems very mysterious to most people. That's too bad, because it can be quite fun and rewarding to write software. To succeed, you just have to take everything in small steps, think very hard, and never give up.

I think everyone studying Information Technology should learn at least one programming language and write at least one program. Why? Programming forces you to think like a computer. This can be very rewarding when dealing with a wide range of IT-related issues from tech support to setting up PPC (pay-per-click) advertising campaigns for a client's web site. Also, as an IT professional, you will be dealing with programmers on a daily basis. Having some understanding of the work they do will help you get along with them better.

Software programs are normally written and compiled for certain hardware platforms. It is very important that the software is **compatible** with all the components of the computer. For instance, you cannot run software written for a Windows computer on a Macintosh computer or a Linux computer. Actually, you can, but you need to have special emulation software or a virtual machine installed. Even with this special software installed, it is still normally best to run a program on the kind of computer for which it was intended.

There are two basic kinds of software you need to learn about as an IT professional. The first is **closed source** or **proprietary** software, which you are not free to modify and improve. An example of this kind of software is Microsoft Windows or Adobe Photoshop. This software model is so popular that some people believe it's the only model there is. But there's a whole other world of software out there.

The other kind of software is called **open source** software, which is normally free to use and modify (with some **restrictions** of course). Examples of this type of software include most popular programming languages, operating systems such as Linux, and thousands of applications such as Mozilla Firefox and Open Office.

But what is the real difference between open source and closed source software? Is open source source software just about saving money? Let's investigate. Let's say for instance you find a bug in the latest version of Mozilla Firefox. The bug is causing a major project to fail and you need to fix it right away. This is not very

likely to happen, I realize, but it's just an example. You might take the following steps:

Step 1. Download and unzip (or uncompress) the source code from Mozilla.

Step 2. Use an Integrated Development Environment (IDE) and a debugger to find and fix the bug in the source code. Please note that you will need to know a little C++ to debug applications such as this.

Step 3. Test the fix and then use a compiler to turn the source code into a binary file. This can take a long time for big programs. Once the source code is compiled then the program should work!

Step 4. You are almost done. Now send the bug fix back to the Mozilla Firefox team. They may even use your bug fix in the next release!

Now imagine you find a bug in a proprietary code base such as Microsoft Word. What can you do? Not much, just file a bug report and hope someone fixes it at some point.

This is a rather radical example, but I think it illustrates to a large degree why programmers generally prefer open source software to closed source alternatives. Good programmers love code and they want access to it. Hiding the code from a programmer is like hiding the car engine from an auto mechanic. We don't like it!

Now you have learned a little about software. You will learn more about software applications and programming in later units.

EVIDENCE 2: GLOSSARY

https://www.english4it.com/module/core/unit/2/vocabulary

abort

to end a program or a process before its completion

When the word processor application crashed, the user had to abort the program and lose all his unsaved changes.

bug

an error or glitch in a computer program caused by a programmer's mistake

An average developer will create one bug for every 10 lines of code written.

closed source

software in which the license stipulates that the user cannot see, edit, or manipulate the source code of a software program

I wanted to develop a new feature for the program, but I couldn't because it was closed source.

compatible

capable of being used without modification

The IBM 360 was the first commercially successful computer family with a wide range of compatible parts.

crash

a computer failure which aborts an application or freezes an operating system

The user was advised to reboot the computer after a serious crash in which the computer no longer responded.

end user

a person who uses a product or service on a computer

Developers must maintain a close relationship with end users if they want to have a successful career.

error

an incorrect action attributable to poor judgment, ignorance, or inattention

The computer reported a "division by zero" error and automatically aborted the program.

execute

to start a program on a computer

The program was set to execute every night at midnight.

feature

something a computer program is "supposed" to do; often a reason to buy or upgrade software

The man upgraded his copy of Word because of a new feature that allowed him to spell-check documents in Spanish.

IDE (integrated development environment)

an application normally consisting of a source code editor, a compiler and/or interpreter, build-automation tools, and a debugger

The new employee asked his boss to buy him a license for his favorite IDE because there was none installed on his new company laptop.

open source

a program in which the code is distributed allowing programmers to alter and change the original software as much as they like

The article stated that many programmers prefer open source solutions because they can modify features and fix bugs without waiting for an upgrade or patch from the manufacturer.

programmer

a person who writes or modifies software

The software company needed to hire three new programmers to help debug their flagship application.

proprietary

privately developed and owned technology

Because of proprietary code, you may not modify or redistribute the source code of Windows or Macintosh operating systems.

restriction

a rule or law which limits or controls access to something

The video website had content restrictions in place for users under the age of 18.

usability

a measure of how easy or hard a program is to use

Back in the day, programmers or salespeople would often be responsible for the user experience, but now we assign that task to a properly trained UX designer.