Spring 5-2020

How User-Friendly Operating Systems and Interfaces Make Technology Harder for Everyone

Cameron Short

How User-Friendly Operating Systems and Interfaces Make Technology Harder for Everyone

Cameron Short

Senior Honors Project

Submitted in partial fulfillment of the graduation requirements of the Westover Honors College

Westover Honors College

May, 2019

Abstract:

This thesis explores both deprecated and current Operating Systems to understand the changes of user interfaces. By comparing older and more flawed Operating Systems interfaces, I intend to demonstrate how user-friendliness has come to the forefront of OS design. However, by building more user-friendly interfaces, the functionality and power of the older designs, while often misused, has been lost. Systems design restricts user access to such functionality because it might damage the system. It is a sensible precaution for users unaware of the effects of certain actions. However, this thesis will argue that for trained and educated users of the system interface modern, user-friendly design limits what users could do with the system. The thesis also includes other examples besides Operating Systems interfaces; similar functionality restriction can be seen in programming languages. C++ required users to allocate their own memory and subsequently delete the allocated memory. If not, errors occurred resulting in bugs of the program. Modern languages like Java and C# have eliminated the necessity for users to manually allocate and delete memory, instead opting for having automatic memory handling. Similarly, C++ pointers were not maintained onto the later dot net (.NET) languages. Overall, this thesis will explore if these changes have been good for the users as well as the programmers. This thesis will attempt to explain why simply educating users about systems, instead of removing useful tools in fear of user error, will make technology better for everyone.

Introduction

Computers are vastly complex machines. It takes a college-level education and maybe even more to understand how they truly work. However, it is ridiculous to expect every person who uses a computer to have a degree. Most operating systems do a good job of making the user's life easier. By hiding things, mostly in the hopes that if an unknowing user doesn't accidentally change something fundamental for the system to operate, then there won't be any catastrophic errors to the system.

Instead of fearing the user and hiding the functionality, it is better to educate the user regarding the purpose of exactly what they were about to change and how the system does not and, in fact, needs them not to change that thing; in short, that system setting it too important to let the user go about changing it when they don't know what it does. That *thing* can be a multitude of operating system components. Learning all these things is a vast undertaking. However, teaching users about computers is a successful venture. It has been done before and can be immensely helpful to the user by showing them how their computer system works.

Background

While they weren't always as complex and easy to use as the ones today, a computer must have an interface that the user can give input to and receive output from. Some of the earliest batch computers had a minimal interface of this kind. Computer programs were "written" on punch cards—the computer code was punched out on them—and the cards were loaded one by one into a computer that would process the cards. After it was finished, the computer would output the results, and they were given to the owner of the program, along with all the encoded punch cards.

Luckily, by the mid-nineties there were plenty of serviceable graphical interfaces. A graphical interface is an interface that utilizes graphics on the computer's monitor screen. One of the most basic tools for an interface is the *Windowing System*. This system separates all of the running processes into different, rectangular sections called "windows" [1]. Each window is a space containing the program so that it can be organized on the screen. Similarly to how papers can be scattered across a desk, with some on the left and others on the right and even some laying on top of others, windows can be organized on screen. Certainly, it's a pretty useful feature to be able to organize all the programs that are open on the screen.

Toolkits and widgets expand the usability of windows. Windows do not inherently contain all the functions that we are used to. The menu bar, scroll bar, any buttons, and other features are all part of a toolkit that offers higher functionality on top of the windowing system. The shift in the nineties was to rely more on virtual toolkits as well as systems that sped development of these interfaces [1]. This shift originated in the fact that, without using these kinds of tools, programming these interfaces "by hand" is difficult and long. Also, coding the same functionalities is repetitive.

Windowing systems provide fundamental support for users of the operating system. And each one, though the look and feel of each is definitely different, shares essentially the same interface features. Both Mac OS and Microsoft's Windows utilize a windowing system. However, there have been many Operating Systems besides these two commercial giants.

Peng, in her doctoral thesis, analyzed and characterized the evolution of operating systems. She categorized all of the studied operating systems based on different intrinsic and extrinsic factors [2]. Examples of her extrinsic factors, which she defines as "factors that describe the outer environment in which the Operating System (OS) exists and evolves and are typically time relevant," include Institutional Support, Governmental Support, Grassroots Support, Organizational Support, and Industrial Support. The Intrinsic factors were more numerous but grouped into categories; intrinsic factors include Resource Management, Design, Cost and Usability, among others.

Unsurprisingly, her statistical models showed that Mac OS, Linux, and Microsoft Windows were consistently among the highest-rated operating systems. While her work is fifteen years old, it is still noteworthy since it accurately predicted three of what have become the most prominent and prevalent operating systems today.

Perhaps contemporary operating systems' success is due to their design. Operating systems with a rich user-experience design should be more successful than those without [3]. Fortier defines a "Rich Experience" as what occurs when the user of a system has an emotional reaction while using that system. A user might feel satisfied and happy when they complete a task made easy by the system. Perhaps that task was not completed easily but was incredibly fast and still successful. Only part of that rich experience lies in everyday use of the operating system. It also includes the customer service of the company, media marketing, packaging, and any other aspect of interacting with the operating system. The Rich Experience that Fortier discusses is meant to refer to any interaction, no matter the relative significance, that a user has with the product, or the company that made the product, or the other customers that also use the product. The ease of using the interface, as well as how that interface relates to the computer's hardware, are a small, but integral part of the rich user experience.

So, the three operating systems that Peng's models determined to be the best probably involve a Rich User Experience design as well. The design is evident in how a system makes tasks simpler and faster or how easy it is to use the interface. As mentioned before, most operating systems' user interfaces are relatively similar, with comparable windowing and widget components, although they might have slight changes. Most of the important changes between each operating systems' GUIs (Graphical User Interface) are about how they're implemented on the hardware of the machine. However, it is fairly evident that both Mac OS and Windows are highly user-experience driven. Consumers of both OSs develop brand loyalties based on which operating system they use. This loyalty is strong enough that most Mac customers will buy the newest model despite how expensive Apple's products are. Windows fans appreciate the personalization and affordability of any machine running the Windows operating system.

On the other hand, Linux has a smaller movement since it isn't as big a commercial venture. In fact, the Linux kernel is completely free and even open-source, meaning that anyone can access the kernel code and modify it if they wish. Unlike Apple and Microsoft, who hide their operating system's code to protect their business, Linus Torvalds and a collective of approximately 25,000 contributors frequently update Linux's online GitHub repository under the GNU General Public License [4]. However, having garnered a base of so many programmers with the single, cooperative goal of improving Linux, it can be sure that Linux is driven with a user-experience design.

Operating Systems Comparisons

Several operating systems were chosen to analyze how they present themselves to users. These operating systems were chosen specifically because of their popularity and transparency. The Linux operating system, being open source and constantly updated, offers a look at one OS that presents its users with a lot of functionality and access to the features that are hidden by other popular operating systems. Windows 7 was chosen due to its recent end-of-life as well as its popularity that lasted for so many years. Windows 10, being the evolution of Windows 7 as well as Windows 8 and 8.1, was chosen to see how Microsoft had updated and changed their OS since Windows 7 released nearly six years before. Windows 8 and Windows 8.1 were skipped due to their relatively short lifespan and general unpopularity.

While the Linux operating system is not just a console-based OS, meaning that it is not just an OS that takes input from a command line where programmers write UNIX commands, it is focused generally on optimizing performance and efficiency. Windows, on the other hand, proudly promotes its ingenuity and user interface. Linux and Windows tend to appeal to different user bases, and it is clearly evident in what each OS's focus is.

Linux

To start the discussion, an introduction to a simple concept might help to distinguish Linux's implementations based on efficiency. Linux prides itself on being cleverly efficient. Examining a routine aspect of the operating systems will provide an insight on where Linux's pride originates from. One of the most common actions users take on a computer is to look at files they've created or downloaded. The file system organizes every file on a computer and is a representation of how that file is stored in memory.

Computer memory is volatile, meaning that if the power is turned off, anything stored in memory is lost forever. The natural extension of memory is on the computer's hard drive, which may be a rotating disk or a solid state drive. Both kinds of hard drive serve the same purpose. The only difference is in the hardware and how it performs the long-term storage. Memory, despite its temporary storage nature, is extremely fast and efficient. Storage is too slow to run the computer off of only the drive. Part of what makes computer memory so fast is how small it is.

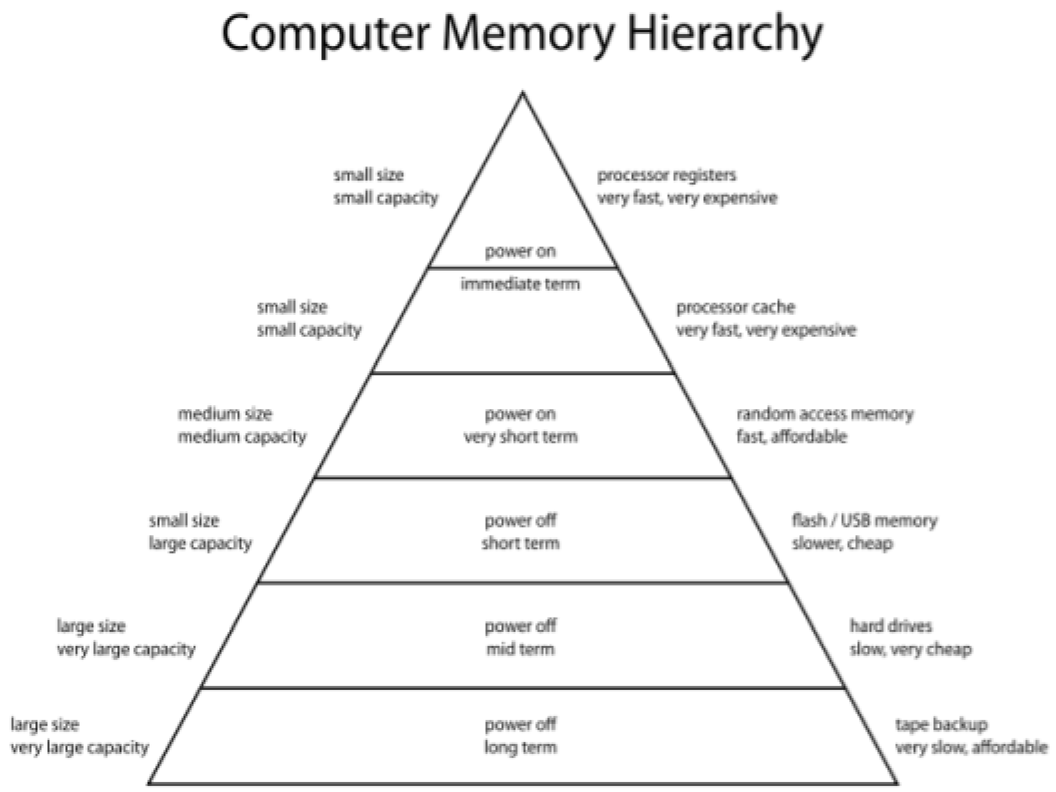

Figure 1. *Shown is the hierarchy of computer memory in relation to size, speed, cost, and longevity. The fastest form of computer memory is the registers that reside on the processor chip while the slowest form are external drives. Registers are tiny and only hold information as long as power remains in the unit. External drives can store information indefinitely with or without power [5].*

When files are sent to long-term storage they must be "swapped" from the processor's registers to the computer's drive. Registers are too small to hold every file in the computer and lose everything when the power source fails or is turned off. So the solution was to create long-term storage that can retain the data. However, the physical hardware of long-term memory is much larger than the fast processor registers [5, Fig. 1] and, therefore, must be located further away from the processor chip. The complex nature of long-term storage as well as its physical distance from the processor makes long-term storage too slow to run the computer. The solution is to transfer, or swap, overflow data to large storage devices from the small capacity registers. This frees the register's limited space for new data.

This size problem becomes a challenge for the OS to overcome as it tries to optimize the efficient use of computer memory when running hundreds of programs both from the OS and those that the user is running. The solution to the problem is breaking the total memory into little blocks called pages. Pages represent a specific amount of memory; pages can be different sizes but are commonly 4K—a tiny four kilobytes—and can be swapped in and out of memory. This swapping presents another problem that is solved by the TLB.

The TLB, better known as the Translation Lookaside Buffer, contains translations for the pages in memory. Memory, like most facets of computer hardware, is controlled by the OS and is often reorganized as time goes on. Running programs are unaware of the OS moving their dedicated memory around and, therefore, don't know where to access the moved memory. The memory address a program may have just tried to access may have been moved to a different location. In order for the program and OS to work together with constantly shifting memory and memory pages, a distinction was made between real and fake addresses. Real addresses are only known to the OS while any running programs that are requesting access to memory do so with fake addresses.

These fake addresses, without going into specifically how the addresses are translated, are used to get to the correct page wherever its real address is currently. The TLB is used to cache these translated addresses so that, when the CPU fetches the address, the process is much faster than if every address is translated each time. However, the TLB is just another kind of memory, and its space is limited. Sometimes, when the OS checks the TLB for an address translation, it doesn't find it and must go through the manual translation anyway.

Linux’s memory management has grown vastly more complicated than its original version and subsequently has been modified to improve its memory handling. One way it does this is by using *huge pages*. A huge page, obvious from the name, is a page that might cover two megabytes or even a gigabyte by using entries in secondary and tertiary page tables which keep track of all the different pages in use [6]. By using these huge pages, some space can be saved in the TLB, making the TLB more reliable. The system is, therefore, faster since it doesn’t have to do as many manual address translations.

Another Linux memory management method is to use Nodes. Nodes allow efficient use of Non-Uniform Memory Access (NUMA) systems. Many systems are NUMA systems, meaning they arrange memory into different parts of hardware on the processor chip. The further away one of these memory locations is from the central processor, the slower it is. Linux treats these long-distance memory locations as a node. Linux thus treats each node as a memory management subsystem to independently control each node [6].

Another common issue computer memory has is that, while fast memory is limited, there are ways to *reclaim* pages that have been given to running programs. However, there is a choice to be made by the OS whether any given page can be reclaimed and made free. Linux does this in a fairly standard way. When the OS allocates too many pages to programs, it will scan through its memory looking for pages of memory that can be reclaimed. If the data in that memory page can be found somewhere else, then the page is freed and marked by the OS as available. However, if the situation is desperate, and a program tries to request a memory page, but there aren’t any freely available, then the OS will actually begin its own special program meant only to reclaim memory pages. Only when there are enough memory pages to satisfy the program’s request will the OS continue running the program.

Compaction is yet another memory management problem. It will be discussed in further detail later, but for now a short description will do. Compaction occurs when, for whatever reason, large amounts of memory are allocated and needed to be contiguous. Remember that the OS is constantly rearranging the pages of the array. However, when compaction is necessary, the OS, similarly to its reclamation program, runs a special program to properly sort all the pages in memory and create as much contiguous free space as possible.

Something that the Linux OS does when it is out of memory to give programs is to invoke the OOM killer [6]. Simply put, the OOM (Out-of-Memory) killer kills a program. It takes a candidate program and kills the program, stopping it from running, in the hopes that after freeing its previously used memory, enough memory will then be available to anything else that needs it.

While this section has been an extremely shallow look at Linux's handling of its OS, it has shown the unknowing user how much is taking place for just the computer memory. The OS handles this and many more issues that arise with a computer's hardware and the requests that are made of it.

Windows 7

Unlike Linux, most of Microsoft's advertisement and appeal lies specifically in its appeal to the average user. Linux is not meant for an everyday user who wants to do simple things on a computer, like browse the web or to write text documents. Of course, Linux can handle that just as well as Windows or Mac OS can, but Linux's appeal lies in high-end server deployment and for professionals who are interested in the efficiency and reliability.

Microsoft’s strengths lie in its design for laymen. Windows 7 is the precursor to Windows 10’s self-purported ultimate usability. The Windows 7 OS and its successors focused on the intuitiveness of the controls and tried to make everything easier for the user to control.

The Windows 7 launch included an upgrade to the taskbar. The taskbar existed before, in Windows XP and Windows Vista, but they didn’t have all of the new functionality that Windows 7 did. The hope was that users would use the taskbar as the primary method of interacting with the OS.

Microsoft dedicated an entire web page just to talk about the updates to Windows 7’s taskbar. The new taskbar keeps all of the functionality from previous Windows versions and adds on to it. The Windows 7 taskbar, which we’ll refer to as just the taskbar, keeps the old Quick Launch feature, allowing a user to click on the taskbar icon and have a program run just from a click. However, the taskbar also included a new feature called Jump Lists [7]. Jump Lists are like program-specific lists of useful links. Jump lists might contain recently opened documents or pinned websites for a browser. Also, if a user hovers over the taskbar’s icons, they can see a live preview of the program. It evolved from showing just a thumbnail of the open program to how the window was actually displaying at the time.

The taskbar was a revolutionary tool for Windows 7. Everything about it screams user-friendly and user-oriented. Combined with Windows 7’s windowing system, this taskbar feature was the epitome of navigating and organizing all of the open windows. Clicking on an open program’s taskbar icon, for example, would bring that window to the front. The fact that it could be customized by the user was another feature of organization. Not only was it easily organized, but this effortless customization built a heightened sense of the rich experience that Fortier talks about [3].

Another aspect of the user-friendliness associated with Windows 7 was for IT professionals. Microsoft released several tools independent of and built into the OS that aided IT personnel in deploying Windows 7. One of Microsoft's updates included a simple but powerfully convenient tool. Windows 7 allowed a user to be able to search not just for files on their own computer, but if a company network or equivalent sharing system had been established, then the user could search through the entire network. Another useful feature for IT personnel was the increased security that Microsoft built into the OS with its BitLocker encryption, which could also encrypt external drives as well as internal data. Microsoft's Desktop Optimization pack also provided a lot of personalization for corporations that were converting to Windows 7 from Windows Vista. So, the Desktop Optimization pack was an incredibly useful tool for IT companies whose clients were concerned about upgrading.

Microsoft definitely focused on the user, but by also making IT professionals' jobs a little easier, it helped improve the overall reception of the OS. Information Technology workers are users of the system too, even if they are the ones deploying and maintaining it. The introduction of Windows PowerShell helped with system administrators maintaining environments remotely by allowing scripts to be written and applied to multiple computers on the system or just to one specific one. This, coupled with a Group Policy Management Console included in the OS's release, allowed IT administrators to easily manage a network of computers.

Windows 7 was truly an operating system meant to appeal to the people. It offered a lot of tools that could automatically handle most jobs that a user might give to it. Both common users and systems administrators received the benefit of Windows 7's feature updates.

Windows 10

Windows 10, being a descendant of Windows 7, has all of its positive features. Some of the old security features like BitLocker are still present in Windows 10 1909, the latest version of the OS. However, there are some newer features that would interest IT professionals [8]. Windows 10 comes with Windows Sandbox, which is an isolated virtual space where users can install software without worry of how it will adversely affect the system.

Of course, Windows 10 also hugely improved upon the Windows 7 taskbar. While the taskbar still exists, it has been coupled with the Start Screen. The Start Screen was introduced in Windows 8 but tweaked for Windows 10. This Start Screen is also fully customizable by the user. While in Windows 7 the user could organize the taskbar only based on what appeared from left to right, the Start Screen allows the user to customize what appears there in a larger two-dimensional space. On the left portion of the start screen is a list of all the programs loaded on the machine that the user can access. Right of the list is an open space where users can pin apps and programs to the two-dimensional space and organize it by named groups. Users can organize this space however they want. They can even change the size and appearance of the program icons.

Generally, Windows 10 seems to be a lot flashier, with an enticing and sleek design. The windowing system's windows all have a sharper, more rectangular look than Windows 7 windows. In fact, it might be said that the design is distracting. When going through the settings, there are many settings and options that are hidden behind links. It's not as if they're hidden from the user entirely, but they are certainly not displayed like the regular settings. These extra, hidden settings are placed on the side of the settings window as a blue hyperlink. When clicked on, they open dialogue boxes with advanced settings to change options most users don't know exist.

This is a perfect example of how modern operating systems hide some features from users. While these "hidden" options are not imperative and most users wouldn't change them if they did know of them, it is still important that they are hidden and only IT personnel would know to find and change them. True to Windows' character, these options are available to the user who wants to change them and, simultaneously, are just out of sight in the spirit of ease in their user interface. Placing these options in hidden menus doesn't overwhelm users who don't want them but still provides access to those users who do.

Modern OS Challenges

Operating systems are not just limited to the overarching and commercially popular giants. There are also many smaller and more individualized operating systems that are built to function in specific circumstances. Operating systems are not just for PCs. Operating systems can be scaled depending on what kind of hardware they run on. There isn't any perfect way to make an operating system either, so the commercial giants who make the most popular operating systems should not be considered the final decider on the subject.

One example is the LEAP OS which Fleming and Adler experimented with running on FPGAs. An FPGA is a Field Programmable Gate Array that is meant to be customized after manufacturing based on the need. Fleming and Adler modified the LEAP OS and took principles from it in order to create an operating system that better suited their needs. Part of their experiment in redesigning the OS for the FPGAs was to see if the LEAP OS could be adapted to their specific system [9].

Another interesting idea is to take virtual machines, virtual spaces running on a computer, that can have a choice of operating system loaded in the virtual space and a choice of how much memory and processing power it has. The idea is to take these virtual systems and make them a cloud-based operating system [10]. Doing so would make the operating system of devices running this new type of OS very fluid. It would make the entire machine programmably changeable, meaning that a few simple changes to settings could have the machine running an entirely new OS without having to reset the device.

Finally, Memory Compaction is an old challenge for operating systems to overcome. Every operating system encounters it, and each chooses its own way to handle compaction. It causes problems with speed since the CPU is stuck waiting on the OS kernel, the brain behind the operating system's special code, to compact all of the pages in memory. Two researchers developed a way to increase efficiency of compaction [11]. Their version, when compared with the standard method, was faster in nearly every measurable way.

It's simple to see that improvements can be made constantly to operating systems and that there is no truly perfect solution. It is also obvious that operating systems are complicated and difficult to implement. The fact that Linux is a free and open-source, community-based platform is amazing. Windows' focus on the user interface and user experience while simultaneously juggling all of the technical interfaces to the hardware is also astonishing. However, all consumer operating systems would be improved if the consumers knew what the computer was doing behind the scenes.

OS's Counterintuitive Design

Each OS, with its respective user interface (UI) teaches users how to do certain tasks in a certain way. Obviously, opening a file from the file system on a Windows machine is a little different from Mac and from Linux. There are different windows and menus and programs and even different file systems for each type of OS. However, none of these OSs teach their users exactly what is going on. Users become familiar with performing the specific task but don't understand how the computer is functioning while their task is being performed, nor do they know what exactly it is that they are doing with the machine. So, when the machine encounters an error or a program crashes unexpectedly the user gets frustrated with the machine.

A comparison of Registers and larger storage, be it a hard drive or external drive, was discussed above. Most files reside on the hard drive of the computer. With the way that the internal architecture of the hard drive is set up, files may be broken up in chunks and split up all across the drive. Having many files get split up across the drive over time can slow down the computer when trying to open and read these files. Imagine you are trying to read a book in the public library, but someone has scattered each page in different sections and different floors of the library. It would take time to search the library for the pages you needed next. The hard drive in a computer functions in a similar way; it must read across different portions of the hard drive to get different chunks of a file.

Continuing the library analogy, imagine if every book had been scattered across the library, mixing pages from one book into others inside the covers of yet another book. It would be a nightmare to try to read a book sequentially from beginning to end. But imagine the librarian occasionally scours the library and puts all the pages back in the right books in the right order. The equivalent of this librarian is the Defragmenting process.

To "defrag" a hard drive means to find all the different chunks of a file and put them in sequential order so that, when the hard drive needs to find it again, it is faster to read. Defragging a hard drive is a relatively simple and routine computer maintenance trick. There are programs that handle the entire process and can even schedule weekly or daily defragmenting sessions for the hard drive. Most users don’t know this information. And yet most users, when taking their computer to a business to fix something, will often ask to “clean up” their PC which often involves this and similar procedures.

The file system is a representation of the state of the hard drive. It’s important to note that even a fragmented drive with scattered files will look the same in the file system as the same drive defragmented. The OS makes sense of all the scattered chunks of files and presents it the same in the UI. This consistency is pretty helpful since the file system is the primary way that users interact with the hard drive. Windows, Mac, and Linux have made the file system very accessible on PCs.

Interestingly enough, it is much harder to access on mobile devices. For example, on iPhones, the file system is virtually inaccessible. There are certain file types that can be accessed through different apps. Image files can be accessed through Photos and text files can be accessed through the Notes app. Looking at one popular forum known for discussion posts about programming and hardware issues among other topics, the only way to look an iPhone’s file system is to either jailbreak the phone or to install third party apps that will present the file system for you [12]. Jailbreaking means to alter the phone by removing restrictions on the device installed by the manufacturer. Usually this means that certain updates will no longer be viable for the device and will not be installed.

Despite the interest in iOS’s file system, Apple has decided to hide it. It’s not as if iPhones don’t have a file system. Any computer that stores files permanently has a hard drive, which requires the OS to handle the file system. Arguably, it does make iOS easier to use without having to worry about the file system. Important file types that users might want to modify are controlled by a parent application which usually provides tools to modify said file. Something can be said about this kind of organization, however it does lead a user's understanding of the device in a less-than-ideal direction.

These two examples have shown how the design of some operating systems has limited the user's understanding of the machine they are working with. An argument can be made that, by making the UI easier to use and by hiding certain details of the computer and OS, users new to a system can easily learn how to use it. Although that is true, teaching the user to properly use the system can eliminate user error and deepen the relationship between technology companies and users.

Computer Education

Not all users need to be Computer Scientists or Engineers; however, understanding the machines they are working with and constantly using should be a low standard. Many users who are older, or just may not be considered 'tech savvy', often use their computers for simple tasks or only for things they know how to do, like browsing the web or writing simple text documents. Instead, all users, of any background and age, should be educated on not only how to use a computer but also how that computer works. It's an important tool to modern life, and users should know what they are using.

Children's education curriculums are always shifting in efforts to teach the most relevant topics in the best ways. Of course, with the turn of the century and Information Age starting, the question of computer education, not just using the computers but learning about them, became a pedagogical interest. In fact, some experiments have been conducted to see if computer education can benefit younger students. Part of this study was done to see if by teaching young students, New Zealand's equivalent of Kindergarten and 1st grade, basic programming concepts it might increase problem solving and logic skills [13]. Both are crucial in Computer Science fields, and so the idea was to see if programming using Scratch Jr., an app that creates game-like basic programming tasks, would naturally manifest these skills.

Accruing data over many hours of video, Falloon [13] analyzed the students' patterns. The students were paired and recorded to see what types of thinking they exhibited while solving their tasks. Not surprisingly, the children's thinking was scattered and distracted in many cases. However, there was evaluative and predictive thinking involved. The pairs displayed the ability to not only try to predict what the code would do, but when it would not work; they also began to debug line-by-line in order to find the error. Of course, the students didn't perform perfectly, as they got distracted or were impatient or afraid of making mistakes. However, it is important to note that they did present the logical thinking necessary for Computer Science-related fields.

While some students in secondary schools are exposed to some programming classes, most are not. Most exposure to the basic programming languages and concepts comes in college and university. Xinagalos studied college-level students to determine several factors like whether starting with imperative programming or object-oriented programming was easier, what language to start teaching in and in what environment to teach it, which strategy the professor teaches the course, and finally, with what extra resources or text books it is taught [14]. Xinagalos' study revealed the difficulties students have when being introduced to programming. It revealed, for example, that students felt they had a difficult time with learning procedural programming but also that they had an even harder time shifting to object-oriented programming.

The programming environments are simply programs that help programmers write code. Code can be written in Word or Notepad, but that isn't as easy as writing it in a dedicated programming environment that helps with error and syntax checking, compiling the different files into one executable, or, simple though it may seem, the overall coloring of the text as it is being written. Small things, like coloring code in certain ways, makes it much easier to read and write code in these environments. The study's concern was whether the environment being taught in was one that was conducive for students and learning rather than for professionals and experts who would need more advanced tools.

Overall, the study did discover some difficulties students encountered but none that were truly groundbreaking. His conclusions were fairly simple: Xinagalos recommended starting with a pseudo-language, a mix of English and programming code that is read like English but is structured like code, carrying out exercises and activities in a dedicated lab-time, and carrying out exams for professors to analyze how well the students understood the material [14]. Overall, the conclusions were fairly basic, and most college-level programming courses would only need slight alterations.

Knowing that the education and teaching practices are fairly sound, it is important to pay attention to the classroom environment. Unsurprisingly, more and more classes are including an online option for students who may not have time to go to class or those who may not live close enough to the campus to attend a class. Computer Science lends itself to online courses so this option is important to consider when dealing with computer education. However, the question is how effective the teaching method can be over telecommunication methods to students who may or may not be traditional college students.

The scope of the Wladis and Hachey's research wasn't very broad, as it was limited to only 3,600 students at a single institution, but its results serve the purpose of curiosity enough. Their study conducted at a Northeastern Community College focused on the results of STEM online-classroom participants. It focused on success rates as well as retention; notably, it found that women tended to do significantly better in face-to-face classrooms, but that men and women performed comparably in online courses [15]. Additionally, those studied who were twenty-five years and older tended to do better online than if they had been in classrooms. Younger, more traditional-age college students did far worse in online classes.

Now, the best practices for teaching students has been outlined, and the effects of what kind of classroom have also been noted. However, there is a final concern for those learning programming concepts who do not intend to go into the field. Similarly to the children who were taught programming concepts in the New Zealand study more for logical and problem solving skills rather than to get them into the field, college students might do the same. In fact, one study of Korean students who were majoring in Humanities at their institution succeeded and did quite well in the Computer Science class they were required to take [16]. Their post-class tests to determine if they had learned computation thinking skills had much better results than the test taken at the beginning of the class. This shows that students are capable of learning computational thinking skills and that it might be used outside of Computer Science and Information Technology fields.

Jeannette Wing is actually the person who coined the term "computational thinking." She defined it as "thinking like a computer science" or more specifically, "thinking in terms of how a computer might solve a problem" [17]. Her driving force was to teach the concepts of computer science in all forms of education. Computational thinking is a way of logical problem solving. Like math, the ideas and concepts need to be taught in a certain progression; one must first learn addition and subtraction before doing complicated equations, and algebra must be learned before doing calculus. Wing's original question was how best to teach the concepts and when to introduce them in the public curriculum. However, as we have seen, the concepts can be learned in multitude of ways and at many different stages in a student's life.

Conclusion

Computers are widely accessible and almost mandatory by today's standards. They function in workplaces, the home, stores, restaurants, and even play a part in cars that we drive. Users come in nearly as many forms with differing levels of knowledge. However, the reality that most users are not formally trained in how these machines work or even how best to use the machine for their individual tasks is disheartening.

There has certainly been progress since the introduction of the Personal Computer. Companies have tried making it easier to use the computer by making their user interface easier to navigate and providing functionality that users most definitely use. Yet it's almost insulting the way that Microsoft hides settings and the efficiency that it certainly has. Operating systems are complex and difficult to make, but Microsoft doesn't seem to make any efforts to have the best side of both worlds. Linux is rewarding in its own way because of its transparent nature and community of programmers who are dedicated to improving every aspect of it, but its learning curve dissuades most recreational users before they begin. Microsoft has the better UI; however, it refuses to treat its users with the understanding of how it works.

Operating systems are big and complicated things. Often times they are customized, like Linux can be, and altered to perform specific functions depending on the hardware it is run on. Still, the very real struggle of learning how to use the most common, universal OSs is something that should be eliminated. Common OS UIs often have tutorial elements that hint at certain functionalities. They pop up as 'help', but they don't appear often. OS design should expand on this tutorial aspect. OS engineers should make a full-fledged tutorial that users of a system could access when they have questions. Like a car's Operator's Manual, which has descriptions of what the blinking lights on the dashboard mean, an OS tutorial could cover different subjects for both savant and inexperienced users. Including such a feature in the OS makes it a better quality product. It also leaves companies' customer service available for those who have critical failures or problems that can't be easily fixed. For users, it seems as though, in a roundabout way, the computer is doing what they want it to. Instead of the frustration that users feel when computers fail to behave as expected, the user can use the tutorial application to learn how to fix the problem. Users will begin to turn to the tutorial in order to begin fixing problems. They will know of a tool to fix their problem, the tutorial, and will start using it to learn the solutions to their problems. Doing so will elevate the general user's understanding so that they can better use the machine instead of simple tasks like browsing the internet or checking their email inbox.

Windows and Apple have begun this process. However, the current embodiment of this tutorial is a sad application hidden away in the OS. In Windows 10 systems, the Tips application is in the programs list. However, when using the Tips app, it does not include a broad depth of information. It includes useful tips that most customers would want to know, yet doesn't include other useful tips that aren't immediately in demand. It includes tips for writing on-screen so that users can take notes when browsing on a web page, and it also includes personalization tips for a consumer to change aesthetic colors for the user's theme. Apple has the equivalent application in its Help menu on the Finder Menu. Instead, the ideal tutorial should be something that the user turns to time and time again. It should be something that the user can open and get help with nearly any question they have when using the system. Having a dedicated application, as opposed to the half-hearted modern standard, would train users to learn how to solve their own problems. It would train users, for lack of better words, to become better users.

Microsoft and Apple do not need to turn their users into programmers. Nevertheless, it is a waste to make such an extensive and in-depth system that goes unused by the customer who buys it. However, it has been shown that it is not a lost cause trying to educate all kinds of users. Anyone who uses a computer can learn how best to use it. Additionally, once companies teach their users how to better use their devices and PCs, a mutual respect will develop to further deepen the relationships between user and corporation.

References

[1] B. A. Myers, "User interface software tools," *ACM Transactions on Computer-Human Interaction (TOCHI),* vol. 2, *(1),* pp. 64-103, 1995.

[2] Y. Peng, "Characterizing the Evolution of Operating Systems." Ph.D., New Jersey Institute of Technology, United States -- New Jersey, 2005.

[3] S. Fortier, "A Systems Approach to Rich User Experience Design." M.Des., Carleton University (Canada), Canada, 2012.

[4] L. Torvalds. "Linux." GitHub. https://github.com/torvalds/linux (accessed Oct. 14, 2019).

[5] "Memory Hierarchy". Wikipedia. https://en.wikipedia.org/wiki/Memory_hierarchy#/media/File:ComputerMemoryHierarchy.svg. (accessed Mar. 25, 2020).

[6] "The Linux Kernel Documentation". The Linux Kernel. https://www.kernel.org/doc/html/latest/ (accessed Feb. 20, 2020).

[7] "Windows 7 Technical Library Roadmap." Microsoft Docs. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-7/dd349342(v=ws.10) (accessed Oct. 14, 2019).

[8] "Windows 10." Microsoft Docs. https://docs.microsoft.com/en-us/windows/windows-10/ (accessed Oct. 14, 2019).

[9] K. Fleming and M. Adler, "The LEAP FPGA operating system," in *FPGAs for Software Programmers*Anonymous 2016.

[10] J. S. Rellermeyer, S-W Lee and M. Kistler, "Cloud Platforms and Embedded Computing - The Operating Systems of the Future," *DAC: Annual ACM/IEEE Design Automation Conference,* pp. 1-6, 2013.

[11] J. Lim and H. Han, "Effective Compaction for Kernel Memory Allocator Using Workload Distribution," *IEEE Transactions on Consumer Electronics,* vol. 64, *(2),* pp. 188-195, 2018.

[12] "Easiest way to browse iPhone filesystem". Apple Stackexchange. https://apple.stackexchange.com/questions/54682/easiest-way-to-brose-iphone-filesystem (accessed Mar 25, 2020).

[13] G. Falloon, "An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad," *J. Comput. Assisted Learn.,* vol. 32, *(6),* pp. 576-593, 2016.

[14] S. Xinogalos, "Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy," *Education and Information Technologies,* vol. 21, *(3),* pp. 559-588, 2016.

[15] C. Wladis, K. M. Conway and A. C. Hachey, "The Online STEM Classroom—Who Succeeds? An Exploration of the Impact of Ethnicity, Gender, and Non-traditional

Student Characteristics in the Community College Context," *Community College Review,* vol. 43, *(2),* pp. 142-164, 2015.

[16] M. Kim and H. Kim, "Effectiveness analysis based on computational thinking of a computing course for non-computer majors," *The Journal of Korean Association of Computer Education,* vol. 21, *(1),* pp. 11-21, 2018.

[17] J. Wing, "Computational thinking's influence on research and education for all," *Italian Journal of Educational Technology,* vol. 25, *(2),* pp. 7-14, 2017.