Operating System Interfaces: A Brief Introduction of the History, Features, and Effectiveness of OS Interfaces

Kyle Reinholt

August 25, 2016

1 Introduction

Before the first OS interface was created, many problems in computing had to be solved. For instance, the ability to handle multiple processes at once. The invention of batch processing and interrupts helped handle these issues extensively and help to open up space for creative design. As the number of computations a computer could calculate grew, the minds of designers and researchers began to expand into new areas. With the advancements that had been taking place in operating systems came the idea of creating interfaces in which to control and interact with the processes being executed. The question was, how should this be done? The inventor of human computer interaction (HCI) Douglas Englebart started answering this question in 1962. He published his work on augmenting the human intellect which showcased many factors that should be considered before designing an interface. He also mentions that an interface (he calls it augmentation of human intellect) would assist in solving many common issues [2]. Part of Englebart's work focused on the importance of making complex tasks more simple. The way to make a task more simple is to carry out the job in a way that is most natural to the user. Eight years later, Englebart invented the computer mouse. This allowed people to interact with computer interfaces in a whole new way. Suddenly, they could associate their sight and touch with the experience of being on a computer rather than just using a keyboard to type. In computing, the way in which people exploit human tendencies and characteristics in a way that allows them to perform tasks more naturally and effectively is through interfacing. This paper discusses a brief history of operating system interfaces, the features that create an effective interface, and a comparison and contrast of GUI/CL OS interfaces.

2 History

In 1973, the first attempt at a personal computer was made. The Xerox Alto had a monitor, keyboard, and a mouse. The interface contained a desktop and a complete graphical user interface. The display graphics were in black and white at this time. The creation of the Xerox Alto interface inspired the design behind many late 1970 OS interface designs. In 1984, Apple released The Macintosh who's interface contained things like a calculator and clock that users could move around the screen as they desired. One year later, the Amiga Workbench OS was released. Amiga contained features like selected and unselected icons, which added to the user's experience of the interface. Stereo sound was a new feature that Amiga contained as well as four colors black, white, blue, orange. It has been said that Amiga was ahead of its time. In 1986, 64-bit operating systems became somewhat common. Windows implemented the ability to overlap windows in 1987. The influence behind Apple's modern GUI's came out in 1989 known as the NextStep OS. Windows took a major step forward by utilizing VGA for display and began taking advantage of graphic capabilities to produce a variety of colors to there displays and programs. In the early 90's, operating system interfaces became a lot more colorful, and featured icon shading. After Windows 95 was released, the importance of

putting an exit box on each window was discovered [3]. The late 90's and early 2000's began the revolution of interfaces we see today. Little by little, companies are perfecting their user interfaces by coupling user experience with advancements in technology. As technology improves, so does the creative space for which a designer has to work in.

3 Features that Create an Effective Interface

The first and most important rule of making an operating system interface is to take into consideration the capabilities of humans and design around these human capabilities. When designing any type of system, the design team must understand the details of the problem space. In this problem space, the human interaction with the system creates the most complexities. The technology must be created in a certain way with a certain number of constraints, with a vast amount of tiny details, in order to compute and process data in a particular fashion. However, before this computed data is shown to a user(general user) rather than a engineer, programmer, or software designer, the data must be filtered so that it is easy or natural to understand (a lot of times, design teams create software spaces to easily create filters or abstractions of data i.e. programming IDEs, APIs). A good interface has user-centered design [1]. Endsley points out 3 core principles to user-centered design.

1. Organize Technology Around the User's Goals, Tasks, and Abilities
2. Technology Should be Organized Around the way Users Process Information and Make Decisions
3. Technology Must Keep the User in Control and Aware of the State of the System

I believe these design aspects are the key to a great operating system interface. You must have the ability to multi-task. You must be able to close a program whenever you want to. Having control of your system, having facilities to organize data the way that best fits your situation, and having the capability to make decisions based on these situations is extremely important for a user.

4 Comparison & Contrast of GUIs and CLs

When evaluating which features of an operating system interface assist in productivity versus hindering it, it is important to evaluate the context of what you are using the system for. Most computer users choose to have a Mac OS X operating system or a Windows OS because of how user friendly they are. Not to mention, most computers are used for business applications or personal entertainment, so they want something that is fairly easy to learn how to use. The less amount of complexity, the more user friendly. User friendly meaning a lot of pictures and graphical representations of the status of programs, systems, and processes. The learning curve for how to operate the system is extremely small. Learning how to make changes to the system must be as close to effortless as possible. You can easily, visually track information about your tasks. You have a lot of control of the system or at least you are under the illusion that you do. For the standard user, these operating system interfaces are great, but for a computer scientist, software designer, engineer, there are some features that they lack. For instance, there are core files in these operating systems that do not have an easily accessible interface. It takes an extremely long time to figure out how to do simple things by navigating through the graphical interface. Both Mac OS X and Windows distributions contain a command shell, much like Linux systems. In fact, Mac OS X is based off of the BSD code-base and has similar commands as Linux distributions. The difference between Mac OS X, Linux distributions, and Window's command lines is the ease of use. I have had all 3 operating systems at one point in my life or another, and the easiest for me to use is Ubuntu. I have also used Ubuntu for the least amount time when compared to the other two. The only issue I have all 3 command shells (from Mac, Windows, Linux) is the vagueness in what happened when a command was successful. Sometimes it takes a long time to understand

the data you are presented with because this data is not filtered for the standard user (that is what the GUI is for). The learning curve for how to use a command line can be pretty steep sometimes, but once you figure out how to type commands instead of navigating through physical representations of the file system, your life gets a whole lot better. I prefer the command line interface because you can run scripts in them. Essentially, you can do 100 different things in a couple of milliseconds whereas if you were using a GUI it would take you 100 minutes. The more time I save doing something, the happier I am. I like to get work done as quickly as possible so I have more free time, the Linux command shell definitely gives you the most power and is very natural to use once you get over the learning curve.

References

[1] Mica R Endsley. *Designing for situation awareness: An approach to user-centered design.* CRC press, 2016.

[2] Douglas C Engelbart. Augmenting human intellect: a conceptual framework (1962). *PACKER, Randall and JORDAN, Ken. Multimedia. From Wagner to Virtual Reality. New York: WW Norton & Company*, pages 64–90, 2001.

[3] WebdesignerDepot Staff. Operating system interface design between 1981-2009, 2011.