Thesis Project Portfolio

Desktop Application Testing: Processes Used for Quality Assurance of Donation Tracking Software

(Technical Report)

An Analysis of the Technological Momentum of Computer and the Ethics of After-hours Contact by Employers

(STS Research Paper)

An Undergraduate Thesis

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Fitzgerald Marcelin Jr.

Spring, 2022

Department of Computer Science

Table of Contents

Sociotechnical Synthesis

Fitzgerald Marcelin

Apr 10, 2022

STS 4600

This sociotechnical synthesis serves to outline the two projects in my thesis portfolio, changes that were made to the final product, and what I learned from conducting these experiments. The initial goal of my prospectus was to answer the research question of how teaching practices for bug and vulnerability testing, as well as industry workflows, could be improved to potentially decrease the presence of bugs or expedite the process of fixing them. By conducting these projects, I hoped to gain a better understanding of current software testing practices in order to compare them to past versions. The technical project portion of this portfolio is titled "First Person Account of Software Testing Processes" and details the time I spent as an intern doing software testing with the desired outcome of being able to compile a roadmap of things to consider when creating testing workflows. This list was to contain advice on matters such as selecting test case tracking software, and how to communicate between teams and as individuals in order to reduce the time taken to resolve testing issues. However, the outcome of this project led me to realize that even creating a roadmap could not guarantee that these issues would be resolved. Additionally, the list of successes and failures within the processes being reviewed for this project instead became a list of subpoints used when offering advice to the University of Virginia's Computer Science department on how the curriculum could be improved to better prepare students for jobs in the industry.

The STS topic section of this thesis outlines the technological momentum of the cell phone and personal computer, and how the changes these devices underwent throughout the decades changed how we communicate with each other. In addition to this, the STS topic section considers the ethics of employers contacting their employees after hours via these technologies for work-related matters. Overall, this project resulted in a general timeline that outlined the progress of the cell phone and personal computer from a microprocessor to the powerful communications devices we have today and how our society has come to rely on them.

These two topics are related due to the initial interest that arose during the course of my technical project, as that was when I began to realize how many mediums and employers had to contact workers after hours, and how this is not necessarily viewed as unexpected by employees. While the nature of my work called for occasional contact after hours, it also opened my eyes to the fact that the devices we are seeking to improve through our work as computer scientists are also what could upset our work-life balance to the point of burnout or job dissatisfaction.

The outcome of the STS section concludes that the technological momentum of the computer and mobile phone led our culture to become reliant on them for work and personal use; despite this however, the use of these devices to contact workers after hours is in fact unethical if the purpose of contacting the employee is to have them do a more than negligible amount of work during personal time.Consequently, when creating singular programs and applications I will now also consider what sort of potentially unhealthy power dynamics and practices I could be reinforcing as I try to affect change in what I do. Additionally, I now also think about what future systems could use something I helped create and possibly viewed as inconsequential as a major component of its design, and how that system will affect societal norms. These are two valuable lessons that I do not believe I would have learned without creating this portfolio.

Desktop Application Testing: Processes Used for Quality Assurance of Donation Tracking Software

CS4991 Capstone Technical Report, 2021

Presented to the Faculty of the

School of Engineering and Applied Science

University of Virginia

By

Fitzgerald Marcelin Jr.

Mar 28, 2022

Abstract

As an intern for Innovative Software Solutions (ISS), I tested a donation-tracking desktop application for nonprofits and churches in order to eliminate bugs, determine how well their software performs, and ensure that the implementation of the program meets the entity's specifications. Test case development and execution followed a basic flow of design, development, execution, and adjustment. The testing process utilized a test case management tool from the Inflectra Corporation named SpiraTest. Additionally, the outcome of the testing process saw a successful initial release to the customer, with future version releases undergoing active development as of the writing of this paper.

Introduction

"Our application crashes on startup." "The client cannot access their saved data after the last patch." "Our software has a file upload vulnerability that we missed." Statements such as these are the last thing software developers wish to hear after their application has been released, or after their updates have been pushed to production. However, the unfortunate reality is that incidents similar to these have happened or likely will happen at some point in any software development process. This prompts the question: What can be done during the development process to aid in avoiding such problems? One solution is software testing.

Related Work

Software testing is described in *Software Testing* as "a process, or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended." (Singh and Singh). Testing was one of the main focuses of my internship.

It should be obvious that the software testing process is not perfect, as evidenced by news articles that inform the public of issues such as software vulnerabilities and application outages. One such application outage occurred near the time of writing. According to *The New York Times* journalist Kate Conger, an outage that took Facebook's services offline in October of 2021 was caused by "an audit tool designed to catch mistaken commands" failing to detect an incorrect command being given to measure network capacity.

In addition to the problems posed by missing bugs during the software testing process, Donald Firesmith of Carnegie Mellon University brings out that there are also issues with the more "human" side of testing such as "test planning and scheduling problems," "stakeholder involvement and commitment problems," "management related testing problems," "test organizational and professionalism problems," "test process problems," "test tools and environments problems," "test communication problems," and "requirements related problems." Matters such as these helped bring into focus the importance of the work I have been tasked with doing over the course of my internship.

Project Design

Due to the potentially sensitive nature of the software undergoing testing, it will not be mentioned by name but will either be referred to as "the software," "the donation tracking software" or "DTS." The first steps in the DTS's testing process surprisingly had nothing to do with testing the software itself. Rather, the initial assignment required reading the specifications and requirements of the software. Upon reading a sufficient amount and becoming accustomed to the intended operation of the software, I reported back to my supervisor for further instruction.

The next step in the process required communicating with the IT department to acquire login credentials for the DTS. Contrary to what I initially believed, this process was relatively quick and painless. I was able to obtain credentials within two hours and was thereafter trained on how to create new accounts and credentials for testing purposes. The types of testing performed over the course of the internship were functionality and regression testing. In order to manage the test cases, I used a software called SpiraTest.

This is where the first obstacle to testing arose. Due to the abundance of features and navigation tabs for the SpiraTest application, finding test cases for the most current release created more confusion than necessary. My supervisor and a few of my coworkers even acknowledged that the software was confusing to use.

For example, if one wished to create new test cases from a release, they would have to first navigate to the "Planning" section of the site. On the planning section, there are "Requirements" and "Releases" tabs. The issue here is that clicking either takes the user to pages that are similar in feel to each other, however, only one of those pages allows the user to generate new test cases. Once the user reaches the proper page, generating new test cases requires selection of each sub-folder of cases that should be included in the tests; however, if the user simply selects the checkbox next to the parent folder, the system adds an additional test case with the name of the parent folder alongside the children within the folder. This issue means that the user must open each parent folder and select the test cases by hand. Alternatively, the user may use the "Ctrl+ Left Click" method to select multiple children within the parent folder. The result of this caused the mere generation of test cases to take longer than it should have, and understandably was a source of minor frustration among those who had to use it.

After generating test cases from the requirements, we were then tasked with running the test cases. Again, SpiraTest proved to be a source of frustration. In order to run the test cases, the user must navigate to the “Testing” tab. Underneath this tab are “Test Sets” and “Test Runs”. In order to run the test cases, the user must go to the “Test Sets” tab, find the recently generated set, and then click on the set to be redirected to its overview page. Afterwards, the user must click the “Execute” button to generate a “test run” after being led to yet another page on which the selection of additional options should be made. Then and only then is the user allowed to begin testing.

However, even after navigating the barriers to test creation, I discovered that the software would automatically log you out if you did not interact with the application for a certain length of time even if you were actively in the process of completing tests. Although it is wise from a safety standpoint to automatically log users out, a software tester may find it extremely inconvenient to need to log back in during the middle of testing. This was especially frustrating in the case of the SpiraTest software because in the process of logging the user out, it would only save the test cases completed and not the “Test Run.” Fortunately for those involved in the process of testing the software, the bulk of the testing difficulties came from the software being used to manage the test cases as opposed to the issues related to stakeholders, management, communication, and scheduling mentioned by Firesmith (2013).

Despite a few unforeseen setbacks, the functional and regression testing of the DTS was completed over the span of two months, with fixes being delivered and product delivery happening within one month following the end of testing. The speed of testing and delivering fixes experienced over the time spent at my internship are not representative of every company’s practices. Obstacles mentioned earlier such as management, scheduling and communication are still potential barriers to testing and should not be disregarded because of this experience.

Results

The results of my learning experience saw the DTS complete testing and get delivered to the client for their use. The client was satisfied with the work done for the initial release and the product is now undergoing development for additional releases.Now that the client has this software, they are able to use something tailor-made for them as opposed to the other accounting software that they found undesirable.

Conclusions

While my time spent testing the DTS during my internship was relatively problem free as it relates to those issues mentioned by Firesmith, I aided in the launch of a new software while learning essential soft skills that will give me insight into how to address some of the issues found with the “human” side of software testing.

Future Work

Potential work that could improve this project is collecting first person records from individuals working at software companies that detail the internal workflow as well as perceived and concrete issues within them. This way, there is a broader range of experiences to consult and use as examples of issues within the software testing process.

UVA Course Preparation and Relevance

UVA courses such as Software Analysis and Software Development Essentials are the classes that best prepared me for my internship.

Surprisingly, over the course of the months spent testing, very little of what I learned at UVA seemed to apply to the process. The courses tended to focus on testing types and testing methods, or thinking of ways to make test cases that test the limits of a specific program. While there was a point during the testing process in which I was asked to attempt to “break” the software, I did not employ the practices of static and dynamic code analysis taught in my Software Analysis course, nor did I utilize the methods of examining branch and line coverage learned during my Software Development Essentials course.

Instead, what I learned from my internship was how to communicate effectively and efficiently within a team, how to create short and detailed reports about the state of testing for those not directly involved, and how to communicate and explain security concerns to developers.

Additionally, I believe that the University can improve its Software Analysis and Software testing courses by dedicating time to teaching students how to test applications, write brief yet detailed reports, and use at least one test case management application to better prepare them for jobs in the industry.

My suggestions for improvements to current courses offered are:

1) Have at least one class period during the semester dedicated to learning how to write and format reports for non-STEM individuals. An example would be writing a one page status report for the CEO of some fictional company;

2) Have a week dedicated to application testing practices as opposed to focusing mainly on how to analyze code. While analyzing code is an important part of the testing process, there is no guarantee that every engineer that graduates from the university will be on the team that analyzes code. Focusing even briefly on testing applications would help students know where to start with testing, making them more prepared to think outside of the box when it comes to attempting to “break” the software; and

3) Teach students at least one industry tool for test case management so they will understand what sort of frustrations may arise when attempting to organize the process.

Implementing these suggestions will help to raise the reputation of UVA’s Computer Science program even higher. Not only will it create more effective programmers, but it will also give those programmers exposure to the world outside of the “bubble” of communicating primarily with engineers in their major for four years.

References

[1] Donald Firesmith. 2013. Common testing problems: Pitfalls to prevent and mitigate. (April 2013). Retrieved November 16, 2021 from https://insights.sei.cmu.edu/blog/common-testing-problems-pitfalls-to-prevent-and-mitigate/

[2] Kate Conger. 2021. *Facebook says its outage was caused by a cascade of errors.*, The New York Times.

[3] Sanjay. K. Singh and Amarjeet. Singh. 2019.*Software Testing*., Vandana Publications.

An Analysis of the Technological Momentum of Computer and the Ethics of After-hours Contact by Employers

STS Research Paper

Presented to the Faculty of the

School of Engineering and Applied Science

University of Virginia

By

Fitzgerald Marcelin Jr.

Spring, 2022

Introduction

The 1970's were a time of rapid innovation in the technology space and gave way to the birth of modern computing. In the time between the beginning and end of the 1970's, the first general microprocessor, Intel's 4004, and the first personal computer, the MITS's Altair computer, were launched (Computer History Museum, 1981). These inventions altered the scheme of the workplace, with large corporate entities no longer being the sole owners of computers, as they were able to be scaled down from large mainframes to smaller than the size of a desk. However, these advancements in microprocessors and devices were a vital aspect of many of the changes to modern day working and living, with the line between work and home life becoming blurred at times, as brought out in "What happened to the border?" (Adisa et al., 2017). This claim is further validated by Federico Faggin, the project leader and designer of the 4004 microprocessor, and his statement that "The emergence of these microprocessors was undoubtedly a turning point for the electronics industry".

This brings into question what can be considered appropriate contact outside of work by those in positions of authority, or even co-workers. In this paper, I will examine the technological momentum of personal computing and communications devices and how they have come to shape new societal expectations with regards to interpersonal communication, and work life balance between the 1970's and 2010's. I will also make a case for a specific year in which groups such as businesses and corporations caused a societal shift towards heavy reliance on computers, and a specific year in which it became commonplace for these groups to contact individuals outside of work via the constant interconnectedness of digital communication. Technological momentum is best summarized as an object or artifact that was once shaped by the society around it reaching a point in its level of influence where it begins to shape or control the society around it instead of the reverse (Hughes, 2009).

This thesis will focus mainly on the personal computer (sometimes referred to as the microcomputer) and mobile phones in addition to giving contextual background on topics such as how humans communicate, foundational technologies such as the Internet, and communication devices that were around during the development of the personal computer and mobile phone, such as the pager. This is to avoid having to discuss technological devices such as the laserjet, ink printer, standalone word processor, CD-ROM, or fax machine, which are also devices that changed how we communicate and transmit data, but could not be used as a way to contact an individual after work hours. This document will later briefly consider the ethics of contacting employees and coworkers after-hours regarding work matters. I will utilize Situational Ethics to conclude that this is in fact unethical and should be discouraged in the workplace.

Modes of Communication and Defining Terms

It is generally accepted that there are four types of interpersonal communication. Oral, verbal, nonverbal and listening. However, we now live in an age where silicon based technology has redefined how we go about using these forms of communication and tasks such as letter writing, talking, and listening can now be done in a digital format in addition to analog.

Consequently, in the context of this document, the term “analog communication” will be used as a blanket statement to refer to forms of interpersonal communication not involving a microcomputer. This means that things such as letter-writing and speaking face to face will fall under the umbrella of analog communication.“Digital communication” in the context of this document will refer to all forms of communication done via a computer or mobile phone. Digital communication can also include the pager, but instances in which the pager is included will be specified within the document. The exception to the definition of digital and analog communication as outlined in this paper is making phone calls. This is because phone calls are technically a form of analog communication, but have undergone a transformation into a form of digital communication with the dawn of new cellular technologies. For this reason, I will not include phone calls under either form of communication and will instead leave it as its own form of communication.

The term "culture" in the scope of this document will mostly refer to the working and private lives and activities of citizens of the United States, as a majority of the milestones discussed in this document came from American businesses."Social group" in the scope of this document will sometimes be used in reference to small groups of individuals with a common interest. The word "commonplace" in this document when used in the context of digital communication will mean that the technology exists in such a way that the amount of users worldwide is greater than the population of the United States. While this does not mean that all users of the technology are from the United States, this will make it more reasonable to assume that a large portion of the country's population has access to the technology in question. "Constant interconnectedness" will be used to refer to a cultural state in which individuals can be contacted outside of the workplace at any time and in any place by someone within the workplace with whom they only share a professional relationship. With these terms clearly defined, it is now possible to move on to analyzing the technological momentum of the devices that changed our society.

Analysis of Technological Momentum

1970-1980

The first microprocessor came about in 1971 with the birth of Intel's 4-bit 4004 CPU, originally designed for calculators (Faggin, 2018). One year after the 4004, the 8-bit 8008 CPU was released (Computer History Museum, 1981). Prior to the development of the microchip, computers were regarded as something that could only be afforded by businesses with enough space to accommodate the large boxes containing computing mainframes. This claim can be supported by the context given in the article "IBM Mainframes" written by the International Business Machines company.

In the same year that the 8008 was released, Motorola gave its first demonstration of a cell phone to the FCC (Old Dominion University, 2015). However, mobile phone technology wouldn't see another significant milestone until 1983, 11 years after the 8008 release.

Armed with the technology needed to scale computers down to a form factor that could be placed on a desk, 1974 saw the release of the first personal microcomputer, MITS's Altair 8800 (Britannica). According to Robert Cringely, also known as technology journalist Mark Stephens however, due to the need for the Altair to be assembled by hand, the main social group that was concerned with the technological advancement of the personal microcomputer were computer hobbyists. These hobbyists were individuals who had the time, patience, skill, and knowledge needed to assemble the Altair and also program it (Cringely, 1996). This is because at the time of the Altair's release, there were no major operating systems for personal microcomputers. Instead, the operator interacted with the device via switches on the face of the device.

Given these facts, one can reason that communication inside and outside of the workplace was still largely analog and phone call oriented, with landlines being the fastest way to contact an individual when not face-to-face. Meanwhile, other businesses related tasks, such as word processing, were completed via typewriter. An exception to this would be pager, which allowed for mostly pre-determined messages to be sent to a user, and later, for users to send an acknowledgement to that message (Morton, 2018). This would also mean that the culture of the United States at large had not yet come to rely on microcomputers for communication or work other than storing data, controlling systems, and carrying out large calculations.

The 1977 release of the Apple II microcomputer, allowed at-home computing to go from a hobbyist's pastime to a way to have access to a microcomputer outside of work (Stein, 2011). With it now being possible for both businesses and private users to have access to computers in their spaces, one could make the argument that this was the beginning of detailed two-way digital communication between work and home, and our culture's shift towards becoming highly dependent on personal computers for communication. Given that mobile phones had not yet become the norm in this time, one may reason that companies could communicate with workers after hours via email or some form of instant messaging. However, at this point, the Internet as we know it now was still in the ARPAnet phase, meaning that widespread public use had not yet become commonplace. Consequently, email and instant messaging had not yet become widely adopted at the time, meaning that something like emailing an employee with a request after hours was not yet possible. This is also without taking into account the cost of an at-home microcomputer for a household. Businesses had the means to purchase an Apple II if they so desired, but the Apple II was still initially intended to target computer hobbyists and wasn't as widely adopted by the industry as it could have been (Cringely, 1996). Therefore, even though both businesses and private users had access to personal computers, communication between work and home was still relatively well defined. Of course, there is the exception of instances where workers had pagers. Alternatively, the business in question may have had the employee's landline number, but even so, there was no guarantee that the employee would pick up the phone.

Between the year 1977 and the year 1981, there were several advancements made in the field of computer software such as the release of VisiCalc; however, because the main focus of this document is to review how personal computers changed communication, I will summarize them as opposed to examining them in-depth. The advancements in question include the creation of GUI's (Graphical User Interface), word processing software for computers, and creation of various other computer applications that allowed for easier use.

1980-1990

In the year 1982, IBM released its Model 5150 personal computer. This computer had one of the first early operating systems, MS-DOS, which IBM had contracted Microsoft to create for them (Cringely, 1996). According to the Computer History Museum, the IBM 5150 marked the first widespread adoption of microcomputers, now to be referred to as PCs, in industry. I would argue that because of this, it is appropriate to regard 1982 as the year in which society began to shift towards a dependence on microcomputers in the workplace as opposed to 1977.

At this time, the Internet was still in its infancy, with it still being referred to as the ARPANet. However one of its major components of today, the TCP/IP protocol, was introduced in 1982 (Zimmerman & Emspak, 2017).

Given that a time has now been established for the year in which microcomputers entered both the private and public space, I will now shift towards discussing the advances in other technologies such as the Internet, cell phone and email until the year of 1990. This is because it is through the Internet that computers are able to communicate with one another and send emails, and cell phones present an alternative method of contact when access to a computer or landline is unavailable. While the years between 1982 and 1990 were by no means lacking in technological advancements, not all of the advancements could be considered significant in the scope of this paper. For this reason, the timeline of events during this decade is smaller and somewhat less detailed than that given for the period between 1970 and 1980.

One year after the establishment of the TCP/IP protocol in 1982, Martin Cooper was credited with developing the first cell phone approved for commercial use.

In the time that advancements were being made in microcomputing, the use of email was also on the rise. However, because the ARPANet had not fully developed into the version we are familiar with today, users of email were widely government and business employees (Stanford, 1999).

In 1983, the Domain Name System or DNS was introduced (Zimmerman & Emspak, 2017). The introduction of this system is what allows modern day users to utilize familiar names with .com, .edu, .gov, .org, and similar suffixes on the World Wide Web to find websites and content as opposed to manually typing in the IP address of the server containing the information they wish to view.

The first phone call on a cellular network was also made in 1983 (Smithsonian, 2021). The development and building of cellular networks are not discussed in-depth in this document for the sake of brevity. However, it is worth noting that cellular phones were still large and bulky objects that had not yet taken on the pocket-sized form we know today.

Even with the introduction of DNS, the Internet was still largely business oriented, with private users likely not having access to it at home. In fact, it was not until 1987 that the number of hosts on the Internet exceeded 20,000 (Zimmerman & Emspak, 2017).

In 1989, Motorola released its MicroTAC cellular telephone, which served as the model for flip cell phones today (Old Dominion University, 2015). In the same year, World.std.com became the first commercial provider of dial-up internet service (Zimmerman & Emspak, 2017).

Although the cell phone was reduced to a pocket-sized form factor in 1989, I do not believe that it would be reasonable to mark this year as the year that the cell phone could be considered a prime method of digital communication by which to contact an employee after work hours. This is because, according to Statista, the number of mobile phone subscribers worldwide was less than 34 million until the year 1993.

1990-2000

The years 1990 and 1991 can be considered two more milestone years in the development of the Internet, and the cultural shift towards constant interconnectedness. In 1990, Tim Berners-Lee developed HTML (Zimmerman & Emspak, 2017), which can be briefly summarized as the computing language that allows users to view content on the Internet. A more technical definition exists, but is not highly relevant in the context of this article.

In 1991, CERN introduced the World Wide Web to the public (Zimmerman & Emspak, 2017). This is arguably the year in which the Internet in its current form was born; an interconnected network of networks used for information exchange, entertainment, commerce, and social interaction. While all of these activities were not immediately conducted on the World Wide Web at its genesis, the next four years saw them come to fruition between 1992 and 1995.

In order to continue moving the discussion forward, it will also be assumed that email became a part of our culture during this time, since anyone with access to the World Wide Web and a PC could create an account with relative ease.

While some individuals may introduce the counterargument that our culture became fully dependent on personal computers and mobile phones after the creation of the World Wide Web in 1991, I argue that it would be best to instead mark 1998 as the beginning of constant interconnectedness and the shift from analog to digital communication in our culture. I would also argue that it is the point at which the technological momentum of the computer and mobile phone began to shape society, as opposed to it being shaped by society. I would choose 1998 as this year because, as mentioned earlier in this article, the number of mobile subscribers worldwide reached a number greater than the population of the United States of America both now and then (Statista, 2021) in addition to the World Wide Web taking on the form we are familiar with today. This means that in addition to being able to communicate with friends and family across the Internet during private time, it would also be reasonable for an employer to have an individual's cell phone number and therefore be able to communicate with them after work hours. This could effectively be viewed as always being connected to work and family no matter where an individual may be or what they may be doing.

2000's and On

One can argue that the 2000's and on have been characterized by what advancements improved upon the culture of constant interconnectedness. FaceBook was launched in 2004 and has gone on to become one of the most influential and sometimes controversial social media platforms. Unlimited phone plans came to fruition around 2003 (Thompson, 2003), and 2007 saw the release of the iPhone, a touchscreen device that most of the population is familiar with.

Technologies such as SMS texting and phone calling have become commonplace with cellular devices, and even personal computers to the point where it is nearly impossible to be unable to contact another person. Furthermore, with the creation of wireless internet and its near constant availability, even personal computers and laptops are a viable method of staying connected in the event that one is unable to find or obtain a cellular phone.

Given our now interconnected society, it would be wise to consider the ethics surrounding contact from employers after work hours. This will be discussed in the following section.

Analysis of Situational Ethics

Situational ethics is defined by the New World Encyclopedia as "a teleological and consequentialist theory of ethics concerned with the outcome of an action as opposed to an action being intrinsically wrong as in deontological theories."

While the current culture of technology has made it possible to contact loved ones and friends at almost any time, it has also made it possible for employers to contact their employees at almost any time. When one takes into account the events prior to and during the 70's with regards to companies being the first entities to adopt computer dependent means of operation, it becomes easier to see how that same shift could cause the lines between work and private life to become blurred. Since operations are largely computer based in our current era, some companies now provide work cell phones and work laptops to enable employees to complete tasks. However, due to the mobile nature of these devices, employees taking them home is not viewed as unordinary. Additionally with the dawn of the Internet age between the 90's and 2000's, most homes now have some form of Internet connection, meaning that these work devices are almost never disconnected from the world at large.

A short search engine query will yield an abundance of articles and stories that talk about instances in which employees have been called into work on a day off or pressured to complete a task by their employer after hours via cell phone or messaging. For example Bryan Robinson, a writer for Forbes Magazine states in one article titled "*It's becoming illegal if employers contact employees after work, new research shows*" that "nearly 70% of workers reported their employer contacts them outside of normal work hours at least once a week. And almost two in five employees said they work outside of scheduled hours because their boss expects it".

According to the same article by Bryan Robinson, 63% of employees were okay with receiving contact after hours during a work emergency and 46.1% were okay with it for urgent deadlines. A work emergency in this context is defined as " an unforeseen situation that threatens your employees, customers or the public; disrupts or shuts down your operations; or causes physical or environmental damage" (Robinson, 2022).

With it now being a cultural norm to contact individuals at almost any time during the day or night, the ethics of contacting employees and coworkers after-hours regarding work matters via personal computer or mobile phone should be called into question.

I would argue that despite constant interconnectedness being a cultural norm, contacting co-workers after hours is unethical via situational ethics if the objective of speaking to the employee or co-worker in question is to request that they complete a work assignment that was not agreed upon or required during work hours. To clarify, I argue that it is unethical for an employer to contact a worker after hours and request that they complete a new assignment that they were not given during work hours or was not clearly outlined to require completion during work hours.

Unless the employee agreed to the possibility of being contacted after work hours in their initial contract (or are in a line of work where there are “on-call” shifts), depending on the power dynamic, the employer or coworker reaching out with the request could also be violating the ethical principle of respect for persons. Respect for persons is best described as one’s requirement to respect the dignity and individuality of others and to avoid using them solely as a means to an end. Robinson’s article points out that 90.4% of employees wanted to be notified during the interview process of how contact outside of working hours is handled. With this statistic in mind, one could argue that if such a high percentage of employees feel that they should be notified of potential after hours contact, it could be viewed as a violation of respect to contact them after hours if it was never mentioned in the initial interview.

One counterargument to this may be that the employer or coworker may be contacting the employee to request that they send a document that may be missing or complete a document. Because of our societal shift towards constant connection in the 90’s and current technological provisions by employers, the employer knows that their employee likely has the time and resources to carry out the task with little difficulty at home. Therefore, the employer may reason that it wouldn’t be unethical via utilitarian ethics to make the request because the time needed to complete the task may be small and they have provided the resources to do it. This is because utilitarian ethics bases how right or wrong an action is based on how it affects the person carrying out the action and the people affected by it. Consequently, due to the nature of corporations, the employee may suffer while the employer and company at large reap the positive benefits, meaning that the net result is viewed as good and ethical via utilitarian ethics.

But this is where the violation of respect for persons comes into play, as this is essentially using the employee as a simple means to an end.

Additionally, because this thesis is utilizing the frame of situational ethics, I argue that this is still unethical because of its principle of putting others before oneself and doing “what best serves love” (New World Encyclopedia, 2022). I argue that the provision of work devices to the employee should be viewed as the employer ensuring that work and personal materials do not mix, and that it would be unethical via situational ethics if the true motive for the employer providing the devices was to ask the employee for extra work after hours. Therefore, by contacting the employee after hours instead of showing them consideration and waiting until they return to work, the employer is acting unethically. Additionally, this does not constitute a work emergency that most employees would be willing to be contacted about and should not be done if after hours contact was not discussed in the interview as mentioned in Robinson’s article. The situational ethics of this matter would dictate that it would not best serve love to do something that a majority of individuals find displeasurable.

Conversely, if the employer or coworker is contacting an employee to request that an employee send or complete a work assignment that the employee stated they would complete by close of business on that day, then it could be considered situationally ethical for the employer or coworker to contact them after hours. This could be considered situationally ethical particularly because that task being left incomplete could evolve into a work emergency, which could be the reason the employer or coworker initially requested the task be completed by close of business. If this is the case, then the employee would be violating situational ethics by not putting others before themselves and honoring their agreement.

However, there are other situations where utilizing situational ethics may not be the best course of action. For example, what if the head office of the employer in question is in a different timezone and they make a request for something from an employee who lives in a timezone where work hours are over? Situational ethics could still be applied, but the consequences of doing so could create a work emergency, which could also violate situational ethics.

Conclusions

I conclude that if anything is to be taken away from this thesis, it is that while it is unethical for employers to contact employees after hours given the constraints outlined above, the constant interconnectedness of our society has made it difficult for it to be practical to always utilize situational ethics when making decisions. But despite this, it should be considered unethical for employers to contact employees outside of work hours with new assignments or work related requests because it would violate the ethical principle of respect for persons when viewed through a frame of situational ethics. Additionally, it may only become more difficult to apply situational ethics as newer technologies emerge that allow us to interact with one another in untraditional ways. It is likely that the last thing the inventors of these devices thought of when creating the technologies for them decades ago was that they would lead to issues such as this. However, given that we know these things now, it would serve us well to view this as a lesson in thinking about what aspects of our society may see a "blurring of lines" as a result of new technologies we create.

Finally, I argue that the technological momentum of the computer and mobile phone led our society to become reliant on personal computers when they were widely adopted by the industry and private citizens in 1982 with the release of IBM's Model 5150 computer; and in turn began to cause it to be shaped by the advancements made in the form and function of these devices in 1998 due to the way they shifted our form of communication from analog and loosely connected, to digital and constantly interconnected.

References

Adisa, T. A., Gbadamosi, G., & Osabutey, E. L. C. (2017). *What happened to the border? The role of mobile information technology devices on employees' work-life balance*. Emerald Insight. Emerald Insight. Retrieved March 28, 2022, from www.emeraldinsight.com/0048-3486.htm.

Bryan Robinson, P. D. (2022, March 2). *It's becoming illegal if employers contact employees after work, new research shows*. Forbes. Retrieved May 2, 2022, from https://www.forbes.com/sites/bryanrobinson/2022/03/01/its-becoming-illegal-if-employers-contact-employees-after-work-new-research-shows/?sh=76961626568a

Computer History Museum. (n.d.). *1981: Timeline of Computer History: Computer History Museum*. 1981 | Timeline of Computer History | Computer History Museum. Retrieved March 27, 2022, from https://www.computerhistory.org/timeline/1981/

Cringely, R. (1996, June 6). Triumph of The Nerds. *Triumph of the Nerds*. PBS.

Encyclopædia Britannica, inc. (n.d.). *Personal Computer*. Encyclopædia Britannica. Retrieved March 29, 2022, from https://www.britannica.com/technology/personal-computer

Encyclopædia Britannica, inc. (n.d.). *Utilitarianism*. Encyclopædia Britannica. Retrieved May 3, 2022, from https://www.britannica.com/topic/utilitarianism-philosophy

Faggin, F. How we made the microprocessor. *Nat Electron* 1, 88 (2018). https://doi.org/10.1038/s41928-017-0014-8

Hughes, T. P. (2009). Technological momentum. In Johnson, D. G. & Wetmore, J.M. (Eds), *Technology and Society: Building our Sociotechnical Future,* 141-149.

Intel. (n.d.). *The story of the Intel® 4004*. Intel. Retrieved April 30, 2022, from https://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html

International Business Machines. (n.d.). *Ibm Mainframes*. IBM. Retrieved March 29, 2022, from https://www.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro.html.

Kim Ann Zimmermann & Jesse Emspak. (2017, June 27). *Internet history timeline: ARPANET to the world wide web*. LiveScience. Retrieved March 29, 2022, from https://www.livescience.com/20727-internet-history.html

Morton, B. A. (2018). *Broadcast for one: Paging and network communication* (dissertation). ProQuest Dissertations & Theses, Ann Arbor.

National Science and Media Museum. (n.d.). *A short history of the internet*. National Science and Media Museum. Retrieved March 29, 2022, from https://www.scienceandmediamuseum.org.uk/objects-and-stories/short-history-internet

Situational ethics. New World Encyclopedia. (n.d.). Retrieved March 28, 2022, from https://www.newworldencyclopedia.org/entry/Situational_Ethics

Smithsonian Institution. (2021, June 16). *The first mobile phone call was made 75 years ago*. Smithsonian.com. Retrieved March 29, 2022, from https://www.smithsonianmag.com/innovation/first-mobile-phone-call-was-made-75-years-ago-180978003/

Spevack, M. (1999). *A short history of email*. The origins of e-mail. Retrieved March 28, 2022, from https://cs.stanford.edu/people/eroberts/courses/soco/projects/1999-00/internet/email.html

Stein, Jesse. (2011). Domesticity, Gender and the 1977 Apple II Personal Computer. Design and Culture. 3. 193-216. 10.2752/175470811X13002771867842.

Thompson, N. (2003, May 23). *Phone companies see their future in flat-rate plans of many services*. The New York Times. Retrieved March 29, 2022, from https://www.nytimes.com/2003/05/23/business/phone-companies-see-their-future-in-flat-rate-plans-of-many-services.html

Prospectus

First Person Account of Software Testing Processes
(Technical Topic)

Technological Momentum of Computers, Testing and Security, 1979-1999
(STS Topic)

By
Fitzgerald Marcelin Jr.
19 November 2020
Technical Project Team Members: Fitzgerald Marcelin Jr.

Signed: [illegible]________________ Date: 11/19/22
Fitzgerald Marcelin Jr.

Approved: ____ Date ___________
Benjamin Laguelli, Department of STS

Introduction

A common problem that plagues nearly every piece of software written is the presence of bugs and/or vulnerabilities. A research study titled “How Do Fixes Become Bugs” states that ”As a man-made artifact, software suffers from various errors, referred to as software bugs, which cause crashes, hangs or incorrect results and significantly threaten not only the reliability but also the security of computer systems.” (Yin et al. 1). The general consensus seems to be that there will always be some bug or vulnerability present in the software and that the best course of action is not to attempt to find each and every issue before release. It seems that software companies are content with having their developers find as many bugs or critical vulnerabilities as possible before release and then either fix them as they are discovered later. Two scholarly articles help substantiate these claims by respectively stating that “Bugs are detected either during testing before release or in the field by customers post-release” (Yin et al. 1) and that “although a located bug is frequently easy to fix, it is difficult to ensure that all the bugs have been removed from a program” (Firesmith). An article states that Associate Professor Gang Tan of Penn State University claims that ‘there is one bug for every 1,000 lines of code’ on average (Buckheit).

At this point, it would be wise to make clear that while the terms software testing and software analysis may at times be used interchangeably, they are in fact not the same thing. Software testing is defined by the International Business Machines Corporation (IBM) as “the process of evaluating and verifying that a software product or application does what it is supposed to do”. Conversely, software analysis is best described as the process of ensuring that the developer’s implementation is in line with the user or customer’s specification and quality standards. The best way to illustrate these differences is with a construction analogy. Software analysis is similar to the foreman ensuring that the building being constructed matches the blueprints. Software testing on the other hand would be similar to inspecting the building for design flaws that could cause it to collapse under stress.

Overall, there are two types of software analysis; dynamic and static. Dynamic analysis requires the tester to be able to run the code in order to view the output, whereas static analysis sees the tester reading through or otherwise analyzing the source code of the application in order to detect bugs and vulnerabilities.

Software testing usually involves functional and non-functional testing. Examples of functional testing include regression testing, and smoke testing. According to an article from California Polytechnic State University, some examples of non-functional testing include performance testing, usability testing, and security testing. There are also additional forms of testing that are not mentioned here for the sake of brevity.

Even with all of these forms of testing, users are still able to carry out malicious actions through vulnerable or buggy code. As mentioned earlier, it is difficult to ensure that there are no bugs in code via the actual testing process. This can become even more difficult when there is no clear and established flow of events for test case generation, testing, and bug reporting.

The goal of this proposal is to address the issues within the bug and vulnerability testing process in order to potentially reduce their presence or aid developers in expediting the process of fixing them. By carrying out the analysis of testing workflow set forth in the technical problem section of this proposal, and studying the evolution of technology, bug vulnerability and software testing as set forth in the STS problem section; I attempt to create a set of internal or company-wide workflow processes and guidelines for general software and vulnerability testing.

Technical Problem

Quality assurance or software testing in Computer Science is often focused on the technical aspect of the job and filled with processes of bug discovery, report generation, and issue resolution that can take days or even months to complete. According to Donald Firesmith of Carnegie Mellon University, this process also “typically only identifies from one-fourth to one-half of defects.”

This presents an issue for the software development process, as it can leave those doing the task feeling burnt out. Reasons for burnout that may be evident to those involved are; there is no guarantee that the bugs discovered will prevent malicious actors from gaining unauthorized access to user information ,or that patching them will resolve a major issue with the performance of the software.

Another reason relates to an aspect of the quality assurance process that may be overlooked; the tools, communication styles and workflows within the discovery and resolution processes can be frustrating or inefficient. Firesmith supports this point by indicating that aside from “test type-specific problems”, there are also “general testing problems” that pertain to the human actors; stakeholders, management, and developers. Examples of issues that would originate from each group of actors are “stakeholder involvement and commitment problems”, “management related testing problems”, and “test organizational and professionalism problems” (Firesmith, 2013). Frustration with communication styles could likely originate from the problems caused by these groups.

Currently, most companies use some form of issue tracking software to report problems, before assigning them to groups for resolution. Another academic document written by Jorge Aranda of the University of Toronto's Computer Science Department, and Gina Venolia of Microsoft's research department looks at the overlap between problem tracking tools and workflow by more closely examining current practices of resolving bugs.

The results of the paper broke down the bug fixing process to five overarching patterns of coordination that each had subcategories describing methods of solving the issue. For example, having a meeting is one pattern of coordination; and two of the subcategories within were "drop by your office", and "air time in status meeting"(Aranda and Veniola, 2009). The definitions of these methods are as follows; "Getting a piece of information, or bouncing some ideas regarding the issue, face to face informally with a coworker in a nearby office", and "the issue was discussed in a regular group status meeting" respectively.

My technical project will take place over the course of time spent doing software testing as an intern. I will utilize the tools and workflows given to me and make a mental note of any issues or "frustrating" matters that arise with the tools or communication. An example of a "frustrating" matter would be multiple employees expressing displeasure with a particular feature in a tracking software . Following the completion of the technical project, I will provide a technical report and analysis in my undergraduate thesis portfolio. This technical report will detail the testing workflow of the software company and successes and failures within the process and propose improvements for the University of Virginia's computer science curriculum to better prepare its students for work in the industry.

STS Problem

In the book *How Users Matter* (Oudshorn and Pinch, 2005, p. 29), it is stated that in "the 1970's, though the main sites of computer power were still in the mainframes and minicomputers found in corporations, in government, in universities, and in the military, computer power had begun to enter the home." The same book goes on to mention that early versions of personal computers were marketed towards those who were " able to assemble the many pieces of electronics that came in computer kits and make them all work". This presented a barrier to entry and use of personal computers, as not all people had the skill or desire required to assemble the device and there was an air of skepticism surrounding the future of computers.

Thomas Watson, President of IBM was quoted in 1943 as saying ""I think there is a world market for maybe five computers" (Shankland, 2006). Furthermore, Charles Tandy, CEO of RadioShack during the 1970's is quoted as saying "Who needs a computer?"

However, there are now over 21.5 billion devices connected to one another across the world (Vailshery, 2021). We live in a time where technology is being made for ease of access, use, and interconnectivity as opposed to being made primarily for research or business purposes. However, because these devices have become more popular and easier to use, the level of caution needed in the software testing process has increased. An article titled *A Short History of Hacking and Data Security Breaches* from Point Park University details the rise of computer popularity and hacking in the 1970's, the regulation of hacking in 1986, the dawn of the internet in the 1990's, and the beginning of data breaches in the 2000's. These factors have in turn given rise to the need for software developers to think more actively about how a user may intentionally or unintentionally break their program or otherwise gain unauthorized access to information. If this trend of decreasing device complexity and increasing user capability continues, there exists a risk of testers becoming overwhelmed by the amount of safeguards and testing practices to consider in order to stop malicious actors.

In this project, I propose that the increasing power and ease of use in modern computing devices has had a major role in shaping the complexity of the vulnerability and software testing landscape between 1979 and 1999.

I will use technological momentum as a framework for this proposal to show that as computers became easier to obtain and use in society between 1979 and 1999, software and vulnerability testing began to require more effort and attention as well. Technological momentum is best summarized as an object or artifact that was once shaped by the society around it reaching a point in its level of influence where it begins to shape the society around it instead of being shaped by society (Hughes, 2009). I will utilize statistics detailing the percentage of individuals with computing devices, and documents describing security and testing practices of the time as well as notable breaches in security.

Conclusion

In order to propose a general internal workflow for software testing and thereby address the issues within the bug and vulnerability testing process, I will 1) provide a report of a firsthand software testing experience, and 2) utilize the technological momentum framework to analyze computer usage, security breaches and software/security testing practices over time. With the insight gained on the issues within the testing process and the trend of increasing complexity of testing due to better devices, developers can begin to address these challenges now. This may in turn improve the implementation of good security and testing practices to guard against malicious actors as new advances are made in the field of computing.

References

Buckheit, Miranda. “Institute for Cyberscience Co-Hire Hunts Security Flaws in Software.” *Penn State University*, Penn State News, 5 Apr. 2018, https://www.psu.edu/news/research/story/institute-cyberscience-co-hire-hunts-security-flaws-software/.

Firesmith, Donald. “Common Testing Problems: Pitfalls to Prevent and Mitigate.” *Carnegie Mellon University*, 6 May 2013, https://insights.sei.cmu.edu/blog/common-testing-problems-pitfalls-to-prevent-and-mitigate/. Accessed 6 Nov. 2021.

Hughes, T. P. (2009). Technological momentum. In Johnson, D. G. & Wetmore, J.M. (Eds), *Technology and Society: Building our Sociotechnical Future,* 141-149.

IBM. “What Is Software Testing?” *IBM*, IBM, https://www.ibm.com/topics/software-testing.

J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories," 2009 IEEE 31st International Conference on Software Engineering, 2009, pp. 298-308, doi: 10.1109/ICSE.2009.5070530.

Oudshoorn, N., & Pinch, T. (2005). *How users matter: The co-construction of users and Technology*. MIT Press.

University, Point Park. *Defending sensitive information: Data security breaches and Cybersecurity*. Point Park University Online. (2021, March 18). Retrieved November 19, 2021, from https://online.pointpark.edu/intelligence/defending-sensitive-information-data-security-breaches-and-cybersecurity/.

Shankland, S. (2006, December 8). *'The world needs only five computers'*. CNET. Retrieved November 19, 2021, from https://www.cnet.com/news/the-world-needs-only-five-computers/.

Vailshery, L. S. (2021, January 22). *Number of connected devices worldwide 2030*. Statista. Retrieved November 19, 2021, from https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/.

Yin, Zuoning & Yuan, Ding & Zhou, Yuanyuan & Pasupathy, Shankar & Bairavasundaram, Lakshmi. (2011). How do fixes become bugs?. 26-36. 10.1145/2025113.2025121.