4-21-2020

Inclusive User Interfaces: A Theory of Design

Katie Marie Goggins

Inclusive User Interfaces: A Theory of Design

By

Katie Marie Goggins

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

21 April 2020

Honors Capstone Director: R. Kevin Preston

Computer Science Lecturer

Student	Date
Director	Date
Department Chair	Date
Honors College Dean	Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

____

Student Name (printed)

____

Student Signature

____

Date

Introduction

Students in the Computer Science department at The University of Alabama in Huntsville benefit from a brief introduction to user interface design in their junior level coursework. In addition, a limited revisit to design theory in Senior Design is all that is required of students prior to graduation. To produce students prepared to compete in the software engineering workforce, more instruction on user interface design must be provided. There are many other areas of instruction in user interface design that could be further explored by the general body of Computer Science students in an academic setting.

A high-quality user interface is an essential component of a software product because it ensures that the customer is capable and informed when interacting with the product. A product is successful when its information and utilities are presented in a clear and accurate manner. The only way to ensure a high-quality user interface is by considering its design elements from the very beginning of the software engineering process. Successfully following the software design process requires good judgement, extensive amounts of forethought, and detailed planning by the software engineering team. A user interface can take many forms, including non-traditional ones such as audio-visual input or complex tactile input from the user. For the purpose of this analysis of user interfaces, the primary focus will be on graphical, interactive user interfaces; however, there will be a discussion on non-traditional user interfaces for the purpose of including differently abled users. The answer to what makes a good user interface can change between decades or even from project to project, which is aggravated by the constant appearance of new technologies that require new strategies of interfacing with users.

The work put into software engineering project is most valuable when many individuals are capable of effectively using the interface with little difficulty. In the workplace, failing to design a well-functioning user interface can have the final product sent back for clarification to developers or could even result in the client finding another group of developers. Industry leaders like Adobe, Apple, and Microsoft regularly publish their opinions on what makes good user interface design. The U.S. Department of Health and Human Services regularly produces guidelines for usability, interactivity, and visual design of electronics. On a review of all sources, there are some main characteristics of what a good user interface should be: consistent, comfortable, informative, and accessible. The design of a user interface is not something that should be thrown together during the last sprint of a project, it is something that should be developed with the user in mind throughout the entire SCRUM or AGILE development process.

There should be an increased focus on the instruction of user interface design in all university programs, particularly those interfaces that require a complex design to achieve maximum functionality. A lack of design knowledge can be a major roadblock for individuals who are just beginning their careers in software engineering. By focusing more on this topic during university, students will have better and more numerous job opportunities in their chosen fields of work while also possessing additional project examples to present when job searching. A literature review of the design of user interfaces, stressing important things to consider during the software development process, and explaining how to make interfaces that the largest number of people can use would benefit students and their future of the department.

An important part of creating user interfaces is making them compatible for individuals with disabilities, those of differing age groups or skill levels, and ensuring technical compatibility with multiple types of devices. Most Computer Science students are at the beginning of their careers, have recently entered adulthood, or both, and they may not consider users outside of their demographic when developing graphical interfaces. A best practice for designing robust user interfaces is to consider different audiences and how they will be accessing a product. Instruction on user interface design is difficult because those skilled with it tend to “find it difficult to externalize their knowledge, and hence design education is forced to reply so heavily on an apprenticeship system of learning" (Cross, 224). Design is a learned skill that can be guided but must ultimately be developed on an individualized basis. Beginning design theory with foundations in the classroom and allowing students to develop their personal styles through research and project will be how this skill is imbued in future classes.

User Interface Design Principles

User interfaces can take many forms, with different styles capable of achieving similar goals. The complexity is scalable, beginning with single sources of input and going to complex graphical physics interfaces dealing with multiple user input streams. One of the most common, oldest, and most simplistic user interfaces in programming is known as the command line. Receiving only keyboard input and being accessible in all operating systems, the command line is a relatively standardized user interface. It is a very accessible interface, compatible with many text to speech programs and capable of having its visibility and language preferences adjusted with ease (Bennet). They are not always comfortable to the user, but the commands are structured in such a way that makes them easy to use. The areas of design that the command lines excels in can be incorporated into complex graphical interfaces, but require extra work and creativity. Graphical programs can be written to open a command-line style interface to receive information from the user in text form instead of interacting with the graphic.

Game design is a subspecialty that focuses on user interface design, with recently published games having confusing or low-quality user interfaces selling poorly and being given bad reviews. Even the smallest inconveniences or bugs can quickly become what a newly published game is known for, making a development company lose millions of dollars. For example, in 2019 Left Alive was criticized for having "imprecise stealth, difficult and unfair enemy AI" in addition to poor "control responsiveness, and mission structure" (Morales). Game design and the entertainment computing field have some of the most highly scrutinized user interfaces, with one interaction potentially being done billions of times by players. A game can also become quickly known for its inclusiveness and ability to be played by more individuals. In the past decade haptic user input systems have become more mainstream, allowing options for individuals with visual, audio or physical disabilities to play new video games. Dota 2 is a challenging game for veteran gamers, but the battles can now be won by almost anyone with enough skill using adaptive controllers (Egliston). Customers of entertainment computing want a challenging game that requires them to learn and develop skills, no matter their level of physical ability ("Accessible Player Experiences").

Customers of software development companies are often other large companies seeking logistical solutions. Software with intuitive and reliable user interfaces like JIRA and Crocagile are making their companies millions of dollars a year due to their success. Flexibility with logistical programs is a trait of the upmost importance, because different teams and project prefer different formats. One benefit to logistically targeted software is that as it becomes more widely used, companies choose to purchase more related products of that company because its employees are already trained in its use. Organization, grouping, task management, and compatibility with cloud storage and file types are important characteristics of logistical software ("Jira Software").

Most companies focusing on software products have specific user interface design preferences. Adobe prefers interfaces that do not detract, nor add to a software product (Babich). Adobe does not want a memorable or branded user interface, preferring its broad utilities and product selection to speak for themselves. Adobe and Apple have both wholly embraced the minimalist UI design trend, along with a myriad of smaller companies. Apple has turned monotone minimalist interfaces into a recognizable brand. Some of the benefits of minimalist style is that a user gets less overwhelmed by on-screen options and the most important utilities are not outshined by excessive features. Minimalist design can fail when the user struggles to be able to access any necessary functionalities within the interface. A tiered interface structure supports novice to advanced levels of skill, keeping each from getting confused while reducing the opportunity for errors. An example of a tiered interface is how many operating systems have a basic settings panel that allows users to navigate to an advanced control panel. Design options like tiered interfaces, search options, and command line style input can be found in both applications and operating systems.

Apple believes that their iOS that should be different from all other operating systems in a measure labeled as Clarity, which explains that "functionality motivates the design" of all their products ("iOS Design Themes"). Designing an interface specifically to showcase the functions of an application is a great basic goal but giving users ways of combining or even adding functionality to an interface should be the end goal. This prevents users requesting minor new functionalities and updates because they can perform a limited amount of development themselves. Microsoft took this route to a degree, due to them producing such a large variety of products, like the Windows 10 operating system and the Office application suite, where they give users power and responsibility ("User Interface Principles"). The Office suite has always been complex due to the variety of needs it is capable of fulfilling. Excel has a modified command line format where users can develop their own algorithms and models, with no need of involving developers. Seeing how the companies with the most success design their interfaces is a good guide for how current students should model their interfaces.

Web development and application development are similar, but their functionality and capacity differences can make early design choices important. Websites can have their customer and product data be supported by a cheap data storage provider, such as Amazon Web Services, whereas applications are standalone unless they have internet or network supported database accessibility. Some applications have multiple storage options; Microsoft Word has option of data storage on a personal Microsoft cloud, a local machine, a network location, or any combination thereof. Web development tends to build a relationship between the developers and the company, due to the iterative pattern of changes and updates. Applications need to be prepared to work indefinitely once delivered to the customer. In the best-case scenario, a company will be able to continue using a custom application for decades with no updates. A well-made application should allow for minor functionalities to be added to it by the user to keep developer interactions to a minimum.

Adobe prioritizes its functionality and expansive array of products more than a specific visual aesthetic. Adobe's user interface design recommendations are detailed in that they discuss general functionality requirements in addition to design methods. User interfaces must be controllable, comfortable and consistent when Adobe oversees their design (Babich). Consistency in a final product's interface presents itself with partial familiarity of other software products – not reinventing the wheel when it comes to navigation and functionality patterns of applications. The aspect of comfort parallels with Apple's trait of clarity with visual user interfaces ("iOS Design Themes"). The logical layout of an application must be clear before a consistent pattern can be derived; users need the ability to intuitively use the application. The visual phrasing of a system should match the phrasing of its user interface and be recognizable to both users and developers (Nielsen). Clarity and capability can sometimes oppose one another, and when complex programming capability is required, more extensive documentation is needed.

The project schedule and requirements are both very important to the design process, providing the scope for what developers can expect their application to become (Cross). Requirements are an essential part of engineering and computing tasks. They must be hammered out well before development begins so that realistic expectations are common between the customer and provider. Cross details the finest component of software development, stating that "objects are a form of knowledge about how to satisfy certain requirements, about how to perform certain tasks." Some customers want the newest and most modern application layouts, even though having complex and detailed animations is not always necessary for a good project. In many cases customers place importance on the fine details; however, making the general functionality of the program work is most important and should be finished first. With new technologies comes new challenges with the development process.

Platform and Logical Flow

There has never been a time in human history where technology is as integrated into everyday life as it is now. This level of integration provides many new advantages and many, if not more, technical complications. Embedded user interfaces need to work accurately and effectively on more types of systems and technology than ever before. Many websites are compatible with various operating systems on different brands of computers, tables, phones, and even some smart refrigerators. This cross-platform compatibility is a new advancement in interface design. Cross-platform compatibility is in line with the priority of allowing as many individuals to use an application as possible.

Many new technologies are wearable and can take in biometric data. The Apple Watch matches Apple product aesthetics but takes in extensive amounts biometric and location data from the user. The challenges posed by these types of interfaces mainly consist of the product's size dimensions, frequent changing between wireless networks, and reliable Bluetooth compatibility. In cases of very small graphical interfaces, keeping display options to a minimum will reduce application clutter. In general, wearable technology brings with it engineering design issues that must be reconciled with user comfort and understandability. Google Glass attempted an entirely new field of wearable technology that failed due to it not solving these issues. User interfaces in developing fields of technology require more robust testing and user feedback to be successful.

For standard user interfaces on a computer or tablet, user input is taken in a limited manner. Keyboard input is the oldest and most widely used data input method, followed by mousepad tracking and icon selection. Having these working correctly is the most important and should be the first priorities. After this stage, gaming controllers, audio input, real time sensor data such as retinal tracking, and motion capture can be integrated. Understanding the limitations and equipment needed to allow for these methods of interfacing with an application can allow engineers to decide what is the best course of action when accepting input from a user.

Navigation within a user interface should be intuitive and comfortable for a user, regardless of by which medium a user interacts with an interface. Having functionality options grouped by type, proximity, functionality, or made searchable through common terms is an important design feature. Utilizing the early design planning stage to decide which method of organization that would best suit a project is an essential step. Workflow designs are also something commonly found in project-planning interfaces such as JIRA and Confluence.

When giving feedback to the user, the software engineer should take into consideration what format of feedback is given. Tones, warning messages, vibrations, bash style text feeds, visible changes to the interface structure, or even a summary report are all possible feedback options, of which all can be substituted with audio feedback if required by the user. Users should not have to rely on detailed action-by-action feedback to successfully use an application. When the user requests an action to occur, the application should immediately perform as instructed. Apple treads a fine line between babying its users and allowing its users to make final interface decisions, saying that an "app can suggest a course of action or warn about dangerous consequences, but it's usually a mistake for the app to take over the decision-making" ("iOS Design Themes"). Allowing educated users more freedom within an application sets them up for success, not failure. Tutorials, strategically placed help boxes, or even a README.txt within the application can be enough for users to understand the functionality of a program. The most important part is having clear and concise instructions, along with reference material and contact information is important if there are major issues. A feedback stream from the customer in the form of a reporting or bug system for larger programs is a beneficial strategy.

Software engineers must make it easily apparent what commands or actions are needed to occur to have a certain outcome to less technical individuals. This ties into the instruction and command set for a user interface, making sure that fixed commands can be correctly used. For example, Linux is much more direct and has more understandable command outcomes than Windows, even if it less intuitive to use. Everything within the computer and network is accessible through precise keyed commands, and the process can be either painless or complicated depending on the logical organization of a system. Give users the opportunity to fix and add to their applications. An emergency cancelation option, broad troubleshooting features, and a force quit application are important to users.

It is difficult to design consistent program interfaces in university when multiple small projects are being assigned. Professors have specific requirements and expectations for interfaces, which prevents students from developing design skills and troubleshooting. Often, students are required to program in a specific language to succeed in a particular task with a standard interface for professors to use for easier grading. This saves time for grading but can impede the growth of software engineering skills in students. Students need experience early in their degree of study in choosing the best language to complete a task. In upper level courses, students will begin a project only to realize that it is impossible to complete in their chosen language or platform. Each year of coursework should include multiple projects capable of showcasing a student's skill and creativity in different forms of media. Upon the culmination of undergraduate coursework, a university graduate will be capable of designing software and interfaces for almost any audience using a wide variety of mediums.

Inclusive Capabilities

Before beginning a software project, an engineer needs to understand who will be interacting with their work. If less technically experienced users will be present, more robust instructions and plain language descriptions of functionality required. This can take extra time but increases user satisfaction and continued purchasing of software products created by a company. Users will also have different amounts of capability to use the software. Allowing the maximum number of users to successfully and painlessly use the software is an important goal. Unless otherwise specified, a software engineer will only need to account for individuals with minor visual, audio, and physical impairments. Accounting for these impairments is important but is not challenging when the project is started with inclusivity in mind.

There is little instruction in higher education as to the design of user interfaces. In the study of software design, a premium must be placed on design methods and capabilities. Due to this, further education is needed for students to create user interfaces that allow individuals with physical limitations, such as poor eyesight, color blindness, hearing loss, and slower response time, more opportunity to utilize the interface. Of course, reasonable amounts of time should be considered for adaptability, not all programs need to be accessible for every user. Making a user interface accessible for most, especially those with considerations that add only a small amount of extra work or a few functionalities. Approximately one in four adults in the United States have a disability, many of which can affect how they interact with a user interface ("Disability Impacts All"). With such a large percentage of people suffering from these common issues, having a user interface that can include them will add satisfaction and usability.

Approximately 4.6% of adults have a disability that impacts how they see the world ("Disability Impacts All"). There are many free to use tools to increase accessibly for individuals with visual impairments, capable of being integrated into software applications. Common solutions are to have a text to audio option or have a setting to increase the variable of text font size within an application. Adding additional color palates, including greyscale, that are agreeable to the most common forms of color blindness, or at least are not overtly clashing can be beneficial. Many people have hearing impairments, 5.9% in the United States, and can benefit from having audio feedback changed into subtitles or text feedback within an application. For wearable devices capable of providing multimodal feedback, a vibration is also an option (Vitense).

One important issue to take into consideration is the physical ability of a user to operate an interface. While 13.7% of the United States has a mobility related disability, many more suffer from tremors or have difficulty with precise movements. Make sure that precise clicking isn’t required for very important decisions and ask for confirmations when making selections that are not easily undone. By implementing these small changes in a user interface, the number of users capable of using the interface can increase by over 25% with little extra work. Unless you are specifically designing something for use by individuals with extremely severe or multiple types of disabilities, there is no need to design for that. Traditional graphical user interfaces are limited, but non-traditional or modified user interfaces can make fantastic inclusive alternatives.

In many cases, command line interfaces are the most inclusive because of the simplicity of reading with audio and only required keyboard usage. Having a command line interface option within your interface can be helpful, account for keywords and specifications. Some braille systems have functionality with command line and can interface with many text-based applications with real time feedback. Polaris BrailleSense is a functional computer with a braille display screen and the ability to hold thousands of applications, books and files. HP Jaws program that implements audio control when using a regular computer. These are two examples of interfaces designed to fill a need, with one having specific hardware and the other being a robust software application. Technologies that allow people access to information and entertainment help people become more self-sufficient and educated while improving their quality of life.

The average individual is more technologically competent than any other time in history; however, not all users are as equally capable. Excel is a good example of a software package that can be used by both experts and as an entry level teaching tool for data processing. It is good practice to have simple options with more complex and detail-specific alternatives that can be accessed in another menu panel or are labeled as a more complex toolset. Younger children tend to be best at picking up new technologies, even though they have the least amount of experience with technology than any other age group. Education on technology and programming is being integrated into early childhood curriculums. Most individuals have a limited amount of experience with technology but having a detailed instruction manual will cover everyone else.

Giving users complete control allows them to learn how to use a software and its interface faster (Babich). A hands-on approach is both how engineers learn how to design interfaces and how the users learn how to utilize those same interfaces. Always prioritize the user and their happiness with the interface during the design process. Hardware based interfaces can be less forgiving and require extensive testing before being released to users, while software application interfaces can be molded to fit their medium. Interfaces are the means by which engineers provide tools and functionality to users, designed to meet their needs.

Conclusion

The skill of designing of user interfaces is a complex skill that requires many opportunities during universities to develop. From the first step of the software engineering development process to the last, user interfaces and the audience must be considered. User interface design at the least should be integrated into freshman to senior level coursework, even possibly being its own class. One method to creating a user interface design course would be to have no language, medium, or OS requirements to allow students to critically think what characteristics will work best.

Hearing from the very largest and most far reaching software and technology companies as to how they develop their products will prepare students for the workforce. Design theory is also very important for students to have a working knowledge of, so that they can begin their careers with their own opinion and styles. A set of suggestions will be created for review by Senior Design and Software Engineering professors at the university, which will set a premise for new requirements as a part of the Computer Science Senior Design project curriculum. A multimedia presentation will be created for distribution and presentation among Senior Design courses in the department to increase developer awareness of user needs. A multimedia presentation will also be created for graduate level software engineering courses on how to solve inclusivity issues with applications. With these presentations, there consists of problem solving and classroom discussion opportunities. User interface design will need to be reviewed on a yearly basis for updates in best practices and as new mediums are invented. This way, trends can be predicted so that the students can be instructed in that interface format before graduating. A high-quality user interface will always benefit software engineers, organizations, society, and the evolution of technology.

Works Cited

"Accessible Player Experiences." Accessible Games. The AbleGamers Charity. 2018. <https://accessible.games/accessible-player-experiences/>.

Babich, Nick. "The Four Golden Rules of UI Design." Adobe, Adobe XD. October 7, 2019. <https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/>.

Bennet, Peter. "Text to Speech." Text to Speech, sourceforge. Web. <http://jampal.sourceforge.net/ptts.html>.

Cross, Nigel. "Designerly ways of knowing." Design Discipline, Open University. p. 221-227. 4 October 1982. Web.

"Disability Impacts All." CDC, US Department of Health and Human Services. September 19, 2019. Web. <https://www.cdc.gov/ncbddd/disabilityandhealth/infographic-disability-impacts-all.html>.

Egliston, Ben. "It's designers who can make gaming more accessible for people living with disabilities." The Conversation, The Conversation US. January 17, 2019. Web.

"iOS Design Themes." Apple Developer, Apple Inc. 2020. Web. <https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>.

"Jira Software." Atlassian. 2020. Web. <https://www.atlassian.com/software/jira/guides>.

Morales, Greysun. "4 Biggest Critical Flops in Gaming from 2019 So Far." Twinfinite, Twinfinite LLC. March 13, 2019. Web. <https://twinfinite.net/2019/03/4-biggest-critical-flops-in-gaming-in-2019-so-far/>.

Nielsen, Jakob. "10 Usability Heuristics for User Interface Design." Nielsen Norman Group. April 24, 1994. Web. <https://www.nngroup.com/articles/ten-usability-heuristics/>

Simon, Herbert. "The Sciences of the Artificial." The MIT Press. p.111-138. 1996. Web.

"User Interface Design Basics." U.S. Department of Health and Human Services. What & Why of Usability. Web. < https://www.usability.gov/what-and-why/index.html>

"User Interface Principles." Microsoft, Windows Development Center. May 31, 2018. Web. <https://docs.microsoft.com/en-us/windows/win32/appuistart/-user-interface-principles>

Vitense, H.S., Jacko, J.A., and Emery, V.K. "Foundation for Improved Interaction by Individuals with Visual Impairments through Multimodal Feedback." Universal Access in the Information Society 2.1 (2002): 76–87. Web.

THE UNIVERSITY OF
ALABAMA IN HUNTSVILLE

David Cook <dac0010@uah.edu>

Fwd: Completed Capstone

Kevin Preston <rkp0001@uah.edu> Tue, Apr 21, 2020 at 9:29 AM
To: Katie Goggins <kmg0029@uah.edu>
Cc: "Dr. Ranganath" <ranganat@cs.uah.edu>, William Wilkerson <wilkerw@uah.edu>, David Cook <dac0010@uah.edu>

Katie,

I approve your submission of the Honors Capstone project.

Please follow the directions in the email from David Cook for final submission.

R. Kevin Preston
University of Alabama in Huntsville
Computer Science Department - Lecturer

---------- Forwarded message ---------
From: **Katie Goggins** <kmg0029@uah.edu>
Date: Tue, Apr 21, 2020 at 9:15 AM
Subject: Re: Completed Capstone
To: Kevin Preston <rkp0001@uah.edu>

Hello!

Thank you so much, I am ready for step 2.

Katie Goggins

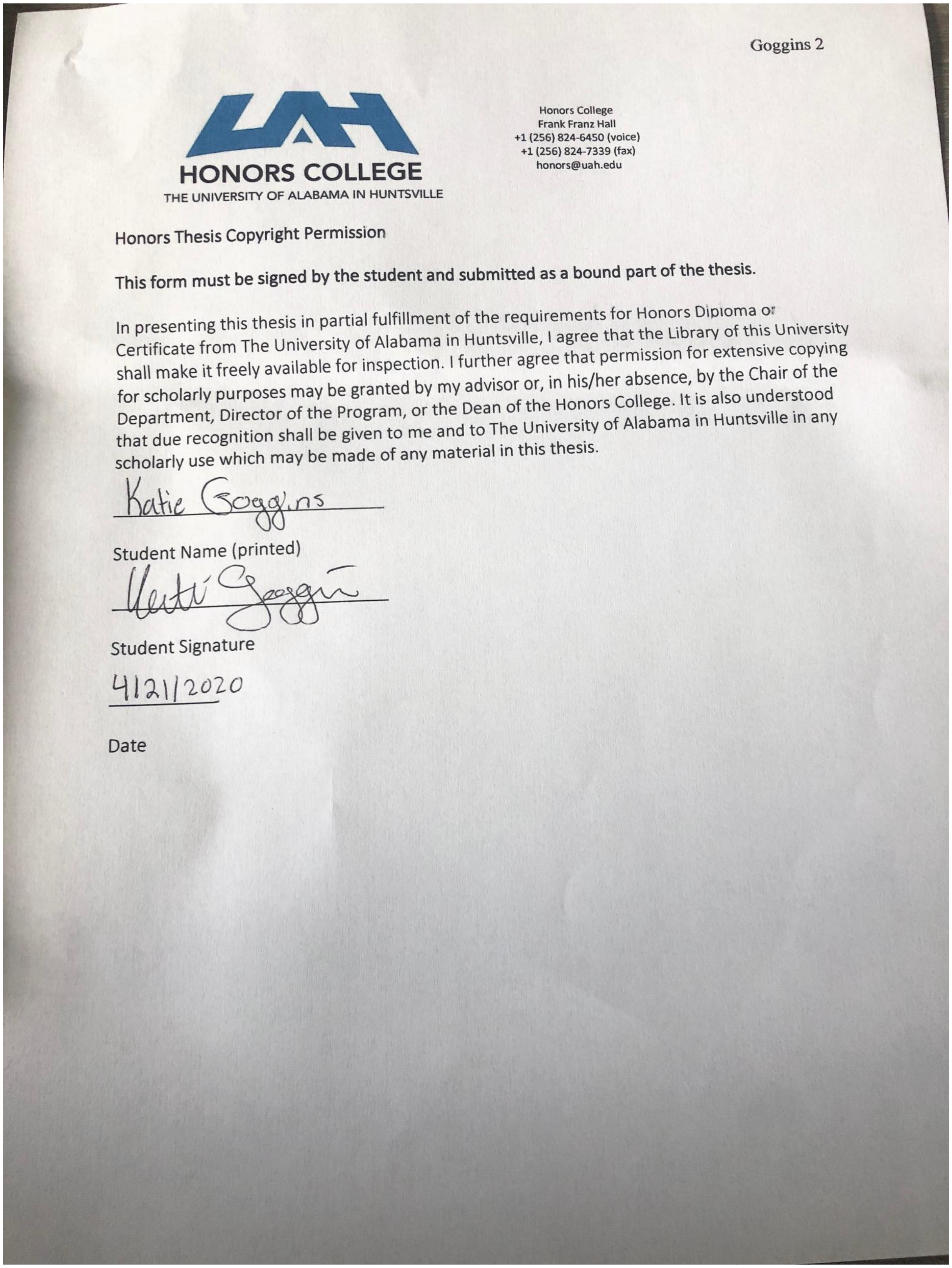
On Tue, Apr 21, 2020 at 7:01 AM Kevin Preston <rkp0001@uah.edu> wrote:

> Katie,
>
> This all looks great.
>
> I forwarded an email to you about how the approval now works. If you are ready I can do step 2 by forwarding your paper to Ranganth and the other approvers.
>
> R. Kevin Preston
> University of Alabama in Huntsville
> Computer Science Department - Lecturer
>
> Virus-free. www.avg.com
>
> On Tue, Apr 21, 2020 at 12:45 AM Katie Goggins <kmg0029@uah.edu> wrote:
>
> > Hello,
> >
> > The revised paper using the comments you gave is attached.
> > The 499 powerpoint is longer, less formal, and has more room to engage with the class than the 650 powerpoint. The 650 is more compact but I believe would still result in many good class discussions.

Please let me know if you want anything changed or added to the powerpoints. Once everything is to your liking, I can sign the first two pages as a pdf and begin collecting signatures.

In other cool news, I was elected last week as the Graduate class senator for SGA, and today was promoted to Speaker of the Senate.

Thanks,
Katie Goggins



HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

Katie Goggins

Student Name (printed)

[illegible]

Student Signature

4/21/2020

Date