The 14th Winona Computer Science Undergraduate Research Symposium

April 28 and 30, 2014

11:00am to 12:30pm

Watkins 108

Winona State University
Winona, MN

Sponsored by the Department of Computer Science at
Winona State University

WINONA
STATE UNIVERSITY
Computer Science Department
http://cs.winona.edu

Table of Contents

Biofeedback: A Player’s Anxiety as Input into a Video Game Environment.

Christopher Bischke
Computer Science Department
Winona State University
Winona, MN 55987
CBischke10@winona.edu

***Abstract*- This paper introduces a hardware-software solution for biological feedback game development. Currently, game developers do not utilize any dynamic inputs that are based off of a player’s biological reactions. The goal of this paper is to research how effectively and efficiently biofeedback can be implemented into a videogame. This process is focused on the hardware and software implementation. On the software side, the game dynamically generate events based on a player’s heart rate. We use a standard heart rate monitor that can be attached to the player’s thumb. Basing the software to respond based on this biofeedback yields an intimate player-game experience.**

***Keywords*- Biofeedback; Video Game; Game Interaction; Hardware; Software Implementation**

I. INTRODUCTION

Video game developers aspire to create more personal and intimate games for players. However, static standard input devices such as a keyboard and mouse vary the experiences that each player has. A third dynamic input device to monitor stress levels of players might improve a game’s ability to create a more personalized experience. The third input would monitor a player’s heart rate. The program would assume a player’s stress level and dynamically craft a unique experience in the game, thus crafting an intimate experience for a player.

Today, gaming is accessible to most of the world, and is more of a common item than it is a privileged item. The Entertainment Software Rating Board (ESRB) is a non-profit body that assigns and standardizes content ratings for video games, so parents may make more informed choices before purchasing a videogame for their child [9]. The majority of the games on the retail market are screened and rated by the ESRB. This means that ESRB is able to collect data from games and consumers. According to the ESRB Video Game Statistics posted in 2010, the average age of a gamer is 34 years old [10]. More importantly, 67% of US households play video games [10]. This data supports that video games are extremely common today and are played over many age groups.

People are interested in gaming, but unfortunately the input devices for gaming have remained rather static since gaming genesis. For home console devices or personal computers: the choice of input has mostly been either a controller with buttons or a mouse and keyboard. Only within the past couple of years have companies such as Microsoft, Sony, and Nintendo experimented with their home consoles with some more dynamic inputs. However, no such inputs have engaged with a player’s personality or biological feedback. Player heart rate monitoring will aid in creation of intentional dynamic game events to sustain elevated player heart rate during play through.

Hypothesis: Player heart rate monitoring will aid in creation of intentional dynamic game events to sustain elevated player heart rate during play through.

II. “FLOW”

One of a game developer’s main goals is to create a game that fully emerges, engages and challenges a player enough to make the game enjoyable [1]. According to the Flow Theory researched by Mihaly Csikszentmihalyi in the early 1980’s [2], video games need to maintain a constant balance between levels of difficulty [2].

“Flow” is described as when the balance of challenge and skill of a player is achieved [2]. In this project’s case, the challenge/anxiety is derived from fear, and the skill/boredom is a product of how frightened a player is. When a player is in “Flow” the game is easy enough to be enjoyable, but difficult enough to be challenging. Ideally, a player wants to remain in flow throughout the entire game, but with static input devices that is not always possible. With aid of a dynamic psychological and biofeedback input such as a heart rate monitor, the game has access to a wealth of player based information.

With static inputs, a developer has to assume what is going to challenge the player. A developer may run a statistical experiment between two game events: A and B. Event A has a more positive feedback, so event A gets implemented into the game. But what about the players that enjoyed event B more? Even if it is a small subset of players, they are not getting as good of an experience as

Event A players. It is possible with enough research into the psychology of videogames along with biofeedback, developers can design games that appeal to a much wider audience.

This research will focus on a horror genre game. When evaluating heart rate of a player, it is easier to assume that a higher heart rate is increased stress, while a lower heart rate is boredom. In order to be able to evaluate other emotions such as, ecstasy, excitement, sadness, and anxiety against each other, the biological input device needs to be more robust than a heart rate monitor. For instance, a heart rate basically has two states, elevated and resting. A developer is going to have to assume and interpret what an elevated state means depending on context. However, a biological device that could monitor electricity signals in the brain can sense an array of feelings, not just two assumed states.

Developers want to keep the player within flow. For the ease and purpose of this experiment, this project is going to maximize the anxiety of the player.

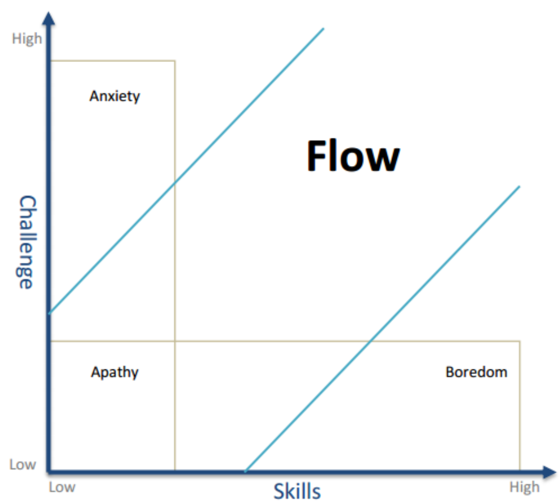


Fig. 1. Mihály Csíkszentmihályi's Flow representation when Challenge and Skills are balanced. Picture provided by [1].

III. HARDWARE

National Instruments supplies the specifications and instructions for a heart rate monitor. This monitor can be built using common items from an electronics store [4].The LED is powered by a voltage source and the LED is faced in such a way that the light pierces the subject's fingertip where the light is received into a photo resistor. The photo resistor outputs a value to an automation I/O device [5] which interprets the data for the game program.

The device is placed on a player's thumb. A disadvantage to this model is that the device is bulky and eliminates the thumb digit, so a player is not allowed to use the thumb digit on the keyboard. A workaround to this solution is to use a more slimming device that would work more like a glove, so all digits can still be used, or use an electro microphone to record the pulse of a player's wrist. However the software does not rely on the hardware implementation, as long as the data is discernable between a calm and stressed state. The photo resistor reads a different voltage level based on the amount of blood flowing through the finger in this experiment's device [6]. This provides for a non-intrusive and easy way to monitor the heart rate of a player [6].

IV. SOFTWARE

This experiment uses VALVe's Source Development Kit (SDK) for the game [13]. VALVe's SDK is a powerful set of tools completely backed by a powerful game engine that has been constantly modified and improved since 2004 [13]. VALVe has successfully released twelve games since the debut of the Source engine. Many of those titles have received numerous awards for their gameplay, mechanics, and storyline. One of the reasons for such highly awarded games is because VALVe is backed by such a powerful toolset like the SDK. The SDK offers robust tools such as a map editor, model poser, and the game engine itself. This project focuses on using VALVe's map editor and source engine for the implementation of this project.

The map editor, also known has "Hammer Editor", is a tool environment that allows the developer to forge the environment or "map" that a player interacts with. Hammer also handles the logic for the world, meaning it handles the creation of game events. In this project's case, the map listens to what events are stressing a player and implements those events later on in the level. The Source Engine is modified to allow Hammer to have an event listener for when a player is stressed.

The game keep tracks of certain game events while measuring a player's heart rate. The code in charge of this is known as the game handler. If the heart rate increases after a certain game event, the game handler remembers the game event for future use. When playing a horror game, if a player is frightened of a certain game event, the game handler remembers that event and then reuse the same elements later in the game to frighten a player again. The game program attempts to keep a player in "Flow".

V. The Game

The game map that the participants will play through is linear style map. Linear means the map has a fix path for the player to follow; however, the game events that generate within the map are completely dynamic. The game generates events based on how the player previously reacted to past events. VALVe previously demonstrated a flow chart similar to the figure below [3].

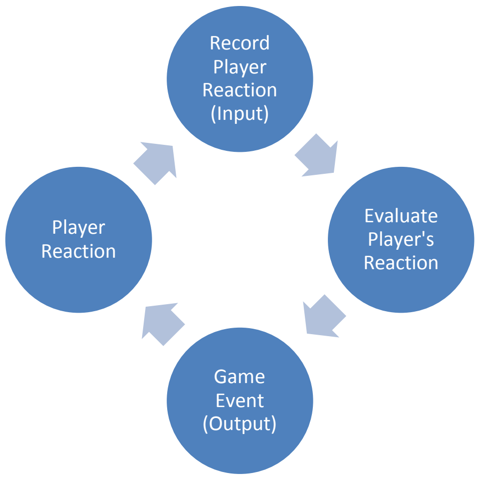


Fig. 2. A version of VALVe's flow chart.

The game generates and host events that are aimed to scare and heighten a player's heart rate. This game follows a similar model to a critically acclaimed horror game called, "Amnesia: The Dark Descent" developed by Frictional Games [14]. Amnesia's gameplay is based around a player being defenseless against perusing monsters. The only way to 'defeat' monsters is to quickly make a decision of where to run and hide to evade opponents. If caught, the monsters kill the player and the game is over.

Most generated events and encounters in Amnesia are based around two premises. First, a planned chase scene: a player is able to see the monster, but the monster is not yet aware of a player. This scenario allows for a player to plan his route and possible actions before he initiates the chase scene. This type of event also allows for anticipation and anxiety to build before the chase scene is initiated. The second type of events are 'jump' events. This type of event is where a player doesn't expect a chase scene and is required to make a quick decision to survive the event. A 'jump' event results in a jump in anxiety or heart rate. The game events are based on either event or a combination of these two type of events.

All the game events that are dynamically generated are being generated for a reason. That reason is based off of the player's biological feedback reaction to certain game events. Either the game is attempting to heighten the level of stress of a bored player or maintain a stress level for a player to keep them engaged. Since the game has a reason for a generated event, the game is able to output that reason for a moderator to view and interpret. This helps with data analysis and understating of the game handler that generates game events.

The creation community for VALVe games is large and supportive. Many map makers and coders release their maps for public use. I used a map created by the username, Riman21 [19]. The map resembles a dirty apartment complex theme. Low lighting and ambient sounds are to increases the overall eeriness theme of the map. When the game begins, the participant is forced into an apartment where static game events are generated. These events are the same to each participant no matter what. The events are jump scene events. Based on how the player reacts, the next event will either be an anticipated stress scene or a jump scene to increase maximum stress.

Once the player has completed the events within the apartment. The player is free to wander a restricted and guided path around the halls of the apartment building. In the halls of the apartment building all the events are generated are dynamic and based off of the reaction of the previous game event.

VI. Implementation

Participants were tested within a confined controlled room. The lights were turned off for the purpose of not interfering with the heart rate monitor and add to the immersion of playing a horror game. A researcher was the only other person in the room with a participant. The researcher's job was to help begin the test, evaluate the stress of a participant, and help if any problems arise. When beginning the assessment, the researcher had the participant sign a consent form that goes over the risks, data, rewards, and purpose of the study. The participant was a complete volunteer and was able to leave at any moment during the evaluation.

Once the consent form was signed, the researcher placed the heart rate monitor onto the participant, and discussed the basic controls about the game. The researcher loaded up an introductory map that has no game events just for the purpose of the player learning game controls. Once the player was comfortable with the game controls, the researcher turned the lights off and started the actual map with the dynamic game events.

While the participant was playing the game, the researcher was evaluating the player's physical stress and the events that were generated during the game. For example, if a participant screams during the game event, the researcher would note the level of stress they assume the player is experiencing. This evaluation did not have to be extremely descriptive since it acted as a small aid later in the evaluation of the biological feedback results.

During the participant play through. The game generated events based on the biological feedback of the participant. Even though game developers want perfect balance of challenge and ease (flow), the game events generated are for maximum scare or stress. The reason for maximum stress is that this study is really focusing on the implementation of the biological feedback through software and hardware. Once an implementation has been proven, further studies can easily work in the psychological flow model into the implementation.

Once the game was completed, the researcher viewed a graph generated by the participant. The graph will display the heart rate data over time, and which events were generated. The game has reasons for which events were generated, and those reasons are based off of the participant's reaction of the previously generated event. By interpreting that data, the researcher surveys what events scared the participant. It is possible that the participant may intentionally or unintentionally gave false information on whether they were scared of an event. So by combining the heart rate results from the video game, and the researcher's evaluation of physical stress during the play through, a confident conclusion may be made on if the correct event was generated for maximum stress. Evaluating the player's stress through survey, physical stress, and heart rate also ensures that the hardware did not malfunction during the test. For example, if the participant clearly demonstrated physical stress during a play through, and the survey confirms the player was stressed – A confident conclusion may be that the hardware failed during the experiment.

VII. Results and Analysis

A total of 20 participants were tested. The number of events that each participant played through were 4 events each. The first event was not dynamically generated, so each participant played through 3 dynamically generated events. So there was a total of 60 dynamically generated events based off of a player's heart rate. Unfortunately, participants number 002 and 003, were not generating numbers that made a lot of sense. Upon further investigation it was extremely probable that the heart rate monitor was broken. For the rests of the tests a new heart rate monitor was issued and the results were more accurate. So participant 002 and 003 are going to be removed from the data analysis because of the monitor fault.

With the two test cases removed, 6 out of the 54 (11%) events were generated incorrectly. Generating an event incorrectly means that the player was visibly scared but the game did not catch the elevation in heart rate. There are numerous possibilities why the program didn't catch a person being scared. One reason could be that not everyone reacts the same to fear or stress. When participants were scared, the graph would represent a large spike in heart rate. However, a small portion of heart rates had a gradual increase in heart rate even though they had physical traits of being scared. The game looks specifically for large spikes in a short amount of time, so if a person's heart rate rises gradually over time, the game will not sense this.

Forty eight out of fifty four (89%) events were generated successfully. To be generated successfully, the game needs to be attempting to scare the player, or continuing to scare the player with the correct event. In the case where a player is bored, the game will alternate and vary the type of events until the player becomes interested again. If the player becomes scared of an event, the next event generated is a similar type of event. Once the player gets bored of a certain type of event, the game identifies the player is bored and generate a new type of event to attempt to scare the player.

Twelve out of eighteen (67%) participants were considered engaged and frighten. To be considered frighten, the game figured out an event that the player was scared of, then kept generating the same event, if the player got bored of the event, the game generates a new event to scare. If the new event continued to scare the player until the end of the game, then the player was fully frightened throughout the game. Six out of eight teen (33%) participants were bored and disengaged throughout the game. This means that the game kept generating different types of events to scare the player, but none of the events raised the player's heart rate. This does not mean that the implementation failed, but may be a reflection of the game design and scare events. By possibly having more events, bored players could eventually be engaged and frightened.

Figure 4 is a graph of a representation of a participant's heart rate who was scared of the first jump event, but then was bored of the second jump event. The game knew the player was becoming bored so generated anticipated stress events for the rest of the game. Figure 5 represents a participant who was scared of all jump events. The game was able to determine the player's fear, so it kept generating jump events. Figure 6 represents the opposite of Figure 5. Participant 010 was not scared of the jump events, but was very freighted and anxious of anticipated stress events.

VIII. Conclusions

Between frightened and bored players, the biofeedback implementation was mostly successful. In general cases, this implementation seems to be a fairly reliable approach. Though, there are instances where this particular biological feedback implementation did not work. First evaluating the hardware: heart rate is extremely context dependent. The game has to know what type of event is being generated and what emotion the player should be experiencing. The player could be react with different emotions than anticipated, but still giving the correct heart rate. For example, a developer wants the player to experience fear at a certain part in the game. The game activates this event and the player's heart rate elevates and the game assumes the player is scared. But the player's heart rate may be elevated for a number of reasons: fear, excitement, exercise, etc. There is no way to confidently say that the player is experiencing the emotions that are intended. A better way to approach the hardware could be by monitoring the player's electrical signals in the brain. Certain signals fire in the brand for certain emotions, so it is a lot easier to assume what emotion the player is experiencing.

This implementation also measures the relative jump of heart rate from one sample to the next sample. This method is advantageous to finding quick jumps in heart rate that are abnormal when the player is calm. This method worked for most of the players, but a smaller set of player's heart rate reacted slower most of the players. This means the participant was notably scared during the play through, but their heart rate just reacted slower to events. In order to accommodate slower reacting heart rates, the algorithm for determining if a player is scared needs to be re worked. An algorithm that can evaluate a player's heart rate over the course of a specific timespan may be a better implementation.

Though there are problems with this implementation. The cost efficiency, ease of implementation, and general effectiveness of a heart rate monitor is suitable for biological feedback in games to invoke and keep certain emotions of the player.

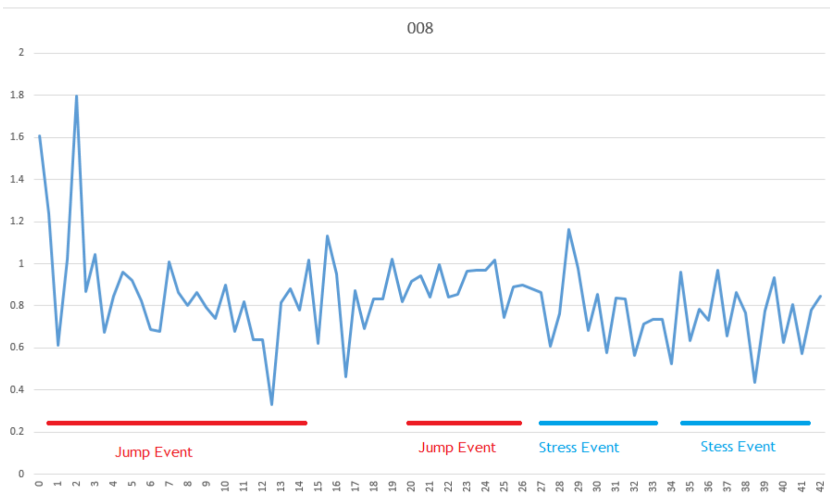


Fig. 4. Representation of the heart rate of player 008.

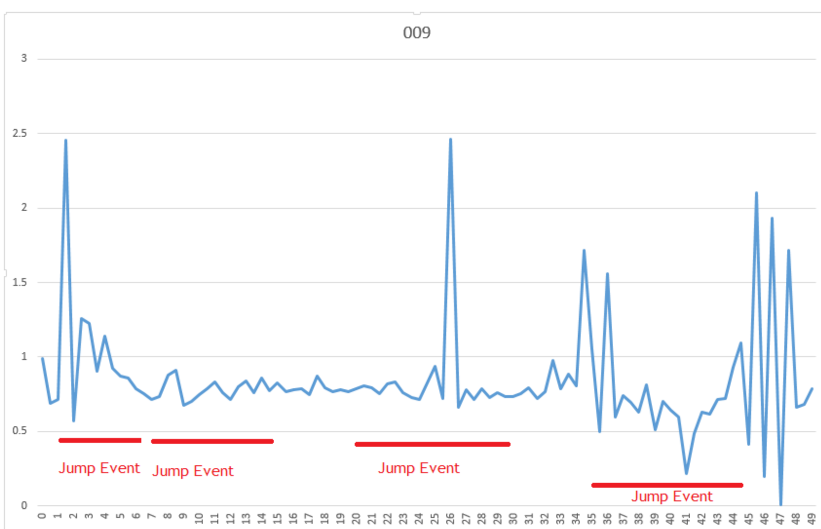


Fig. 5. Representation of the heart rate of player 009.

Fig. 6. Representation of the heart rate of player 010.

IX. Acknowledgments

The author of this paper would like to extend a warm thank you to April Dawn Valete from the University of Minnesota for the aid in the creation of the survey implementation. The author would also like to thank Dr. Debnath, Dr. Iyengar, and Dr. Zhang from Winona State University for helpful advice and guidance during the length of the project. The author would also like to thank Dr. Cichanowski from Winona State University for the critical review of this paper.

X. References

[1] L. Nacke, “Measuring players' experience of games and real-time simulations”, iTEC08 - Your Technology Hotspot, Hessen-IT, Darmstadt, Germany, 2008.
[2] B. Cowley, D. Charles, M. Black, R. Hickey, “Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment”, Volume 6, Issue 2, Article No. 20, April/June 2008.
[3] M. Ambinder, "Biofeedback in Gameplay: How Valve Measures Physiology to Enhance Gaming Experience”, Game Developers Conference, March 2011.
[4] National Instruments, “Build Your Own Heart Rate Monitor”, http://www.ni.com/white-paper/14248/en/, August 2012. Accessed December 2013.
[5] LabJack, LabJack’s U3 Platform, http://labjack.com/u3.
[6] Embedded Lab, “Easy Pulse Sensor Overview”, http://embedded-lab.com/blog/?p=7336. Accessed December 2013.
[7] VALVe Software, http://www.valvesoftware.com/.
[8] G. Fonte, “Pulse Speed Timer”, Nuts and Volts, pp 39-43, April 2007.
[9] ENTERTAINEMENT SOFTWARE RATING BAORD, http://www.esrb.org. Accessed December 2013.
[10] ENTERTAINMENT SOFTWARE RATING BOARD, “How Much Do You Know About Video Games?”, http://www.esrb.org/about/video-game-industry-statistics.jsp. Accessed December 2013.
[11] STEAM, Steam & Game Stats, http://store.steampowered.com/stats. Accessed December 2013.
[12] Microsoft, Earnings Release FY12 Q1, http://www.microsoft.com/investor/EarningsAndFinancials/Earnings/Kpi/fy12/Q1/detail.aspx, 2012. Accessed December 2013.
[13] VALVe Software, Source Engine Platform, http://source.valvesoftware.com. Accessed December 2013.
[14] Frictional Games, Amnesia: The Dark Descent, http://www.amnesiagame.com. Accessed December 2013.
[15] K. L. Amon, A. Campbell, “Can children with AD/HD learn relaxation and breathing techniques through biofeedback video games?”, Australian Journal of Educational & Developmental Psychology, Vol. 8, pp. 72-84, 2008.
[16] M. Nei, Cardiac Effects of Seizures. Epilepsy Currents, Vol. 9, Issue 4, pp. 99-95, July/August 2009.
[17] C. Bischke, N. Debnath, “Hardware Solutions for Biofeedback and Game Interactions”, 2013.
[18] D. Carr, D. Comotis, F. Glass, “Video Game Invasion: The History of Global Obsession”, Beantown Productions, Documentary, March 2004.
[19] Riman21. Dirty Apartment. GameBanana. http://hl2.gamebanana.com/maps/177693. Accessed December 2013.

Improving Navigation Efficiency of Internal Structures With the Use of Augmented Reality

Andrew Dean
Department of Computer Science
Winona State University
Winona, MN
adean10@winona.edu

***Abstract*— First time navigation within a new enclosed structure, such as a school or hospital, can be difficult. Posted signs and maps are the common method of navigation inside a building, though signs are stationary and afford limited amounts of information. Without the use of infrared or near-field communication devices such as Bluetooth or radio-frequency identification (RFID) tags, an exact position cannot be established. Global positioning systems (GPS) are not an option as there is not a clear path between the satellite and receiver. A proposed solution to these problems is the use of augmented reality (AR) and quick response (QR) codes. These can be used to display a detailed map of an internal environment, as well as the user's current location and intended destination. This project used AR displayed on a mobile device (laptop on a cart with webcam) and scanned QR codes in the environment. The user input the destination; directional information was then superimposed atop the QR code giving the user a map with their current position and a path to the specified destination. This combination eliminates the need for near-field communication while giving the user more detailed information than posted signs and maps. The AR software used in this project was the open source program AR-media. A survey was conducted on 30 students at the Winona State University Rochester campus. The survey compared navigating a building using the AR setup verses posted signage. Navigation time and the subjects' preference were the variables measured. The results of the survey have been evaluated to determine if the difference in time and preference between navigation styles is significant.**

Keywords—augmented reality; navigation; QR code; mobile device; localization;

I. INTRODUCTION

Today's advanced technology allows us to get from point A to B easier than ever. The gadgets that assist us with navigation are numerous and complex, but getting started means knowing where you want to go in the first place. GPS devices are meant for outdoor navigation and cell phones that use assisted GPS are only as accurate as the signal they receive to triangulate a location. Inside a building GPS connectivity is not an option and even with cellphone and Wi-Fi reception, a smartphone cannot give you an accurate location.

Without a device to assist with internal structural navigation, directional information is not always displayed in a convenient form. Unless a person knows the layout of the building or campus, the signage is not always appropriate or easy to decipher. In a common situation, the stress of being late to a class or appointment can cause a person to panic and get lost. Augmented interactions have the potential to affect users' psychological and emotional states. For example, augmented interactions with nature – like direct interactions with nature – may help to reduce stress and benefit psychological functioning. [1] This is why I propose a localized augmented reality software solution to direct anyone around inside a new building or campus. Augmented Reality provides a way to present any computer-generated information on the top of a real world. [5] Most augmented reality (AR) systems for indoor navigation are based on the assumption of continuous localization of the user and require either a significant effort to instrument the environment with the necessary infrastructure, or sensor-based estimates of user movement in the environment. [4] The augmented reality setup for this project will overcome this problem by simply using a web camera and software compiled and run on a laptop. In designing augmented reality systems, it is often essential to implement a tagging (ID) system to make a link between physical and digital spaces. [6] The augmented reality setup for this project will scan a quick response (QR) code in a building and provide a location that is assigned to the code. The QR codes can be easily made by normal printers, can be attached to almost any physical object, and can be recognized by mobile readers.[6] Once the code is scanned, the user is presented with a detailed map and a path to follow. The camera attached to the laptop tracks the QR code and changes the orientation of the map displayed, wherever the camera is pointed. The QR code will also display the appointment information related to the surroundings, such as room numbers or building names. The overall objective for this project is to test the effectiveness of an augmented reality mobile setup for navigation within a large building.

***Hypothesis* - The use of the augmented reality navigation setup will decrease travel time and will have higher satisfaction ratings from subjects compared to posted sign navigation.**

II. METHOD

The AR navigation setup was tested by a survey. The survey consisted of 30 current students at Winona State University Rochester. Each student independently navigated to the same location. Half of the students were instructed to navigate to the destination using standard signage, while the others were instructed to use the augmented navigation setup. Several development steps were necessary to set up the survey, including AR software utilization, QR code creation and map association for navigation.

Software utilized in this project included an existing plugin called AR-media and Sketchup, a CAD tool developed by Google. The plugin is a free open source download that can be compiled to run on a variety of devices including Windows, Mac, iPhone and Android. The hardware used in this project was an HP EliteBook 8470p, Logitech 9000 Pro web cam, and a Pryor Products Light-duty Laptop Stand.

The layout of the hardware began with attaching the laptop to the top of the laptop stand with Velcro. This allowed the laptop to be removed when not in use as the augmented reality setup. The web camera plugged into the laptop via USB. There was no attachment of the webcam to the laptop or cart. This allowed free movement of the webcam as if the user was holding a smartphone. Moving the cart was possible due to the five wheels that swivel independently of one another. This and the handle that wraps around the cart, made it easy to push the cart in any direction. To compensate for the various heights of students, the cart was adjustable with a pneumatic piston located between the wheels and laptop surface.

The AR-media and Sketchup software was installed on the Windows 7 operating system. Free trial licenses were applied to both applications to allow basic functions needed to run the setup. Sketchup was installed first to allow AR-media plugin to associate with the toolbars in Sketchup's design environment.

Setting up the augmented reality with AR-media began with creating a quick response code. Typically QR codes are a high definition matrix of squares arranged in a tight grouping. These more complicated QR codes are scanned as a still image and the algorithm used to process the image is more time consuming. The QR codes used with augmented reality are larger, simpler combinations of squares or shapes. These less complicated QR codes are processed in real time by the lower resolution video sensor. For the purposes of this project, a simple white square within a black square and the AR-media logo (fig. 1) was used as the QR code.

Fig. 1 QR code design

Associating the navigation map to the QR code began with importing an image of the floor plan from each building on the UCR main campus. The floor plan images were affixed to the three dimensional plane in Sketchup, with the center of the image set to the XYZ coordinates (0,0,6). This makes the map on the laptop screen appear to float six inches above the QR code when scanned. (fig. 2) Having the map offset from the QR code also prevents the map image intersecting with the QR image. A dashed line was also drawn onto the map to indicate the path for the participants to follow. The line begins from the starting room and follows the optimum path to the destination. (fig.3)

The QR codes were scanned with an external web camera connected to a laptop. This allowed the student to move the camera freely when inspecting the map. The laptop was attached to a cart. (fig. 4) Having the cart mobile enough to transport throughout the campus was taken into consideration for the tests. The cart chosen is widely utilized at a major medical center in the Rochester community. The students scanned QR codes posted in the same areas the building signs were located. There were five buildings used in the navigation path, all connected on the main floor. Each building was assigned a QR code with one being placed at the intersection of adjoining buildings. Each QR code gave the student an overview of their location on campus. The current location of the student was displayed according to the location of the QR code. As participants from the AR group followed the path between buildings, they had the option of scanning a QR code if they felt they needed more information.

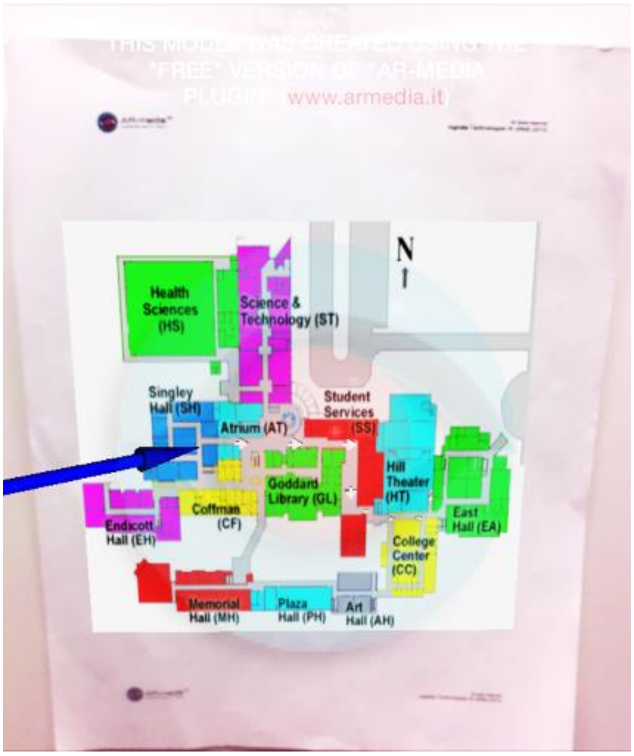


Fig. 2 Map displayed on laptop with AR-media and QR code

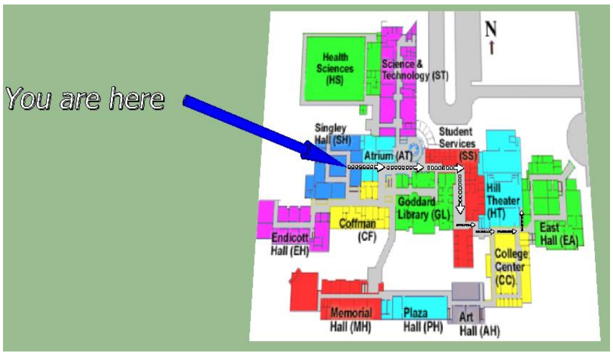


Fig. 3 Map design as displayed in Sketchup

Fig. 4 Laptop attached to cart with webcam

The students were tested separately from each other to prevent copied behavior from one student to the next. Each individual pushed the cart regardless if the AR navigation was assigned to them or not, equalizing the physical demands. The path that the students took began at the computer science department lab, room 101 in Singley Hall and ended in the Hill Theater green room. The end location was picked as it does not have a room number associated with it, except on the detailed floor plans. Each student was followed and timed during the navigation trial from start to finish. No assistance or hints were given to the students during this time. Once the student completed the course the survey was taken. The survey consists of 3 questions:

1. Did you already know the location of the green room?

2. On a scale of 1-5, 1 being very unhelpful to 5 very helpful, how would you rate the navigation method assigned to you?

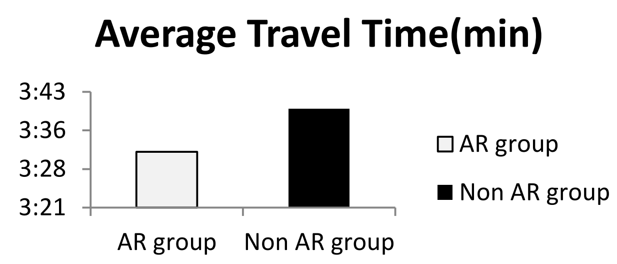
3. Additional comments?

Question 2 is a Likert scale from 1 to 5 that quantifies the students' opinions.

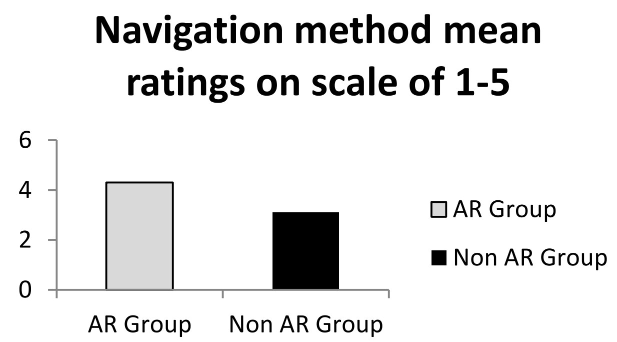
III. Results

Timing from the student trials indicated that there was little difference on average between the two groups. There were outliers from both groups that were faster at finding the destination. The AR group outliers spent more time pushing the cart through the crowded hallways. The non-AR group outliers already had a good idea where the room was located. Of the 30 participants, 28 said the "green room" in the Hill Theater was unfamiliar to them.

The average times for both groups are compared:



The ratings for both navigation methods are compared:



The average time for the AR group was 3:32. This was 8 seconds faster than the 3:40 it took for the Non-AR group to navigate to the same destination. There were 2 outliers that knew the room location were able to walk straight to the room in under 3:00. The strides and heights of the students were not taken into consideration when timing.

The survey results stated that the AR group rated the AR navigation at 4.3 out of 5. Students from the Non-AR group rated the standard sign method of navigation at 3.1 out of 5. The difference in ratings between the two groups indicated that the AR group liked their navigation method 24% better than the Non-AR group.

IV. Analysis

The results from the navigation timing were as expected. Generally, the time it took for both groups was not significantly different. One major difference in the navigation timing was the path that the students chose. Like in turn-by-turn navigation, the instructions were user-centric, and the results of a user study show that it is a viable solution to help users navigate in a building, in the absence of continuous localization. However, the solution is not robust against users' deviations from the pre-defined path, due to the complete lack of localization.[3]

The survey results proved more positive than I expected. Even with using the cart and the camera, the concept was well received by all of the students. The common response from the survey was to implement this on a smartphone. After the student trials with the laptop cart and webcam, I was able to get a working version of the AR navigation on an iPad and iPhone. This was valuable to demonstrate a working version of the AR navigation at the Judith Ramsey Research Seminar along with the poster. The response from the poster was positive, particularly the implementation of the AR navigation in hospitals and large campuses with multiple buildings.

Separate of the testing method, using the laptop on the cart was intended to simulate the pushing of an IV pole or wheelchair while manipulating the navigation software. This helped to support the idea that AR navigation could potentially be used in the healthcare system. The ability to hold onto the cart or wheelchair while navigating would be important for patient safety. An idea brought up by my Professor, Dr. Zhang, would be the implementation of this navigation software on Google Glass. This would allow the user to operate completely hands free.

Other variations of how the AR navigation could be implemented would be to include elements of the surrounding architecture or pictures in the building. Working with augmented reality in architectural rooms is more than providing better and more complete ways of interacting with digital technologies, it further more includes an important task in making the resulting artistic effects of augmented reality an integrated part of the total perception of the architectural rooms and spaces. [2] Beyond using QR codes, the WSU logo could be used to map a location on campus. Pictures on the wall or art in the hallways of a hospital could be used as landmarks that the AR navigation could recognize and display a location.

Determining the success of this project relied on the feedback I receive from participants in the survey. Most of the feedback was favorable with suggestions on how to improve my application. Ultimately I hope to get more feedback on this pilot project to make it into a fully functional application for patients, students, or anyone, navigating a new environment.

V. Conclusion

In this paper we covered the plausibility of using augmented reality as a medium to navigate inside a building. Requirements for the project were discussed as well as the changes that AR navigation could bring. The methodology for implementing the project was covered and results analyzed. The results showed pointed in a positive direction, implying AR navigation can assist people navigating in a new environment. Feedback from the surveys and people interested in the project will help to shape the future design of the project. The current version of the project is a fraction of what would be needed to implement this in an entire building. Databases and additional programming will be needed to realize the full potential of AR navigation on a mobile device.

Acknowledgment

I would like to thank my professor, Dr. Zhang and the students of CS495 for the assistance given to me in writing this paper. The materials and time allowed to work on this project while at work would not have been possible without my employer, Mayo Clinic. Also, love and appreciation to my wife Andrea who contributed her experience with technical writing.

References

[1] Friedman, B. and J. Peter H. Kahn (2000). New directions: a value-sensitive design approach to augmented reality. Proceedings of DARE 2000 on Designing augmented reality environments. Elsinore, Denmark, ACM: 163-164.

[2] Krogh, P. G. (2000). “Interactive rooms: augmented reality in an architectural perspective”. Proceedings of DARE 2000 on Designing augmented reality environments. Elsinore, Denmark, ACM: 135-137.

[3] Mata, F. and C. Claramunt (2013). Augmented navigation in outdoor environments. Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Orlando, Florida, ACM: 514-517.

[4] Mulloni, A., H. Seichter (2011). Handheld augmented reality indoor navigation with activity-based instructions. Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services. Stockholm, Sweden, ACM: 211-220.

[5] Pyssysalo, T., T. Repo, et al. (2000). CyPhone: bringing augmented reality to next generation mobile phones. Proceedings of DARE 2000 on Designing augmented reality environments. Elsinore, Denmark, ACM: 11-21.

[6] Rekimoto, J. and Y. Ayatsuka (2000). CyberCode: designing augmented reality environments with visual tags. Proceedings of DARE 2000 on Designing augmented reality environments. Elsinore, Denmark, ACM: 1-10.

Performance Comparions of Operating Systems for the Raspberry Pi

Joshua C. Dinndorf
Department of Computer Science
Winona State University
Winona, Minnesota
JDinndorf10@winona.edu

***Abstract*—The Raspberry Pi is very popular computer platform that is small, the size of a credit card, and only costs $35. Because of this it is obviously limited in its hardware performance. It is very important to utilize that power as efficiently as possible. The operating system has a major influence on how those resources are used throughout the system. We need to find an operating system that can utilize the Raspberry Pi's hardware to get the greatest performance out of the given hardware configuration. In this paper we look at two operating systems that run on the Raspberry Pi, namely the Raspbian and the Pidora. We compare the performance of these two operating systems in three categories; process management, memory management and secondary storage management. These categories are evaluated using synthetic benchmarks to simulate real use of that service. Results suggest that Raspbian outperforms in one category, process management, while Pidora performs better in secondary storage management and the third category, memory management was inconclusive. Thus we cannot conclude that either of the operating systems will perform better on the Raspberry Pi in all scenarios.**

Keywords—Raspberry Pi; Raspbian; Pidora; Operating System; Benchmark; Performance Comparison;

I. INTRODUCTION

The Raspberry Pi, shown in Figure 1, is credit-card sized computer that cost $35.00 [1]. Obviously at this price point and size the Raspberry Pi (RPi) is limited in what it can do from its hardware. With this limited performance it is very important to manage and utilize it as efficiently as possible. Operating systems play a big role on how resources are used throughout the system.

Fig. 1: Raspberry Pi Model B

An operating system basically is a program that manages both computer hardware and software for the end user [2]. Most computers need some sort of OS, including the RPi. Operating systems are designed to work with specific systems and architectures. The RPi uses the ARM11 (ARMv6) architecture, which is dated and only a handful of operating systems are designed or still support running on it. Of these RPi OS's there are some specialized for certain tasks, like RaspBMC which designed for media centers. Others ARMv6 OS's are either not stable or not optimized specifically to run on RPi. The two compared for this paper are Raspbian and Pidora. These operating systems are both based on major Linux Distributions, they are multipurpose, and they are both stable and have some optimization for the RPi. Since OS's are similar it begs the question which one is actually better suited for the RPi?

In this paper we compare these two operating systems by measuring the performance of the common operations that OS's are responsible for. Operating systems are responsible for a large amount of tasks. These can generally be broken down into three categories. These categories are process management, memory management, and secondary storage management [3]. Operating Systems can implement these categories differently, which can impact the speed and efficiency of the system.

We need a way to compare each of these categories quantitatively so we can evaluate how well each of these OS's utilize the RPi's hardware. This is where benchmarks come in. In general, benchmarking is a way of measuring a task. For computing, benchmarking is a way to measure variables, such as speed or bandwidth, when executing a computing task [4]. This allows comparison between different hardware and software configurations, which in our case is different operating systems. In this paper we use benchmarks to evaluate the categories noted above. We then use the results to compare the two operating systems. The benchmarks we use will simulate real world workload on the specific components of the operating system. It is important to note that while benchmarks give a great view into performance, they do not measure subjective judgments such as user interface or available software. Those preferences are out of the scope of this paper.

The primary motivation to compare these two operating systems on the Raspberry Pi was because of the lack of quantitative benchmark data doing so. There are many articles comparing OS's of the Raspberry Pi on personal option but none of performance. The Raspberry Pi being such a popular computer with such limited hardware it is important to compare the operating systems available for this device on a purely objective, quantitative way and not just on subjective preference.

The next section outlines some related works. After that is a section on the background information on the two operating systems. Then we will go in-depth into our methodology of evaluating the operating systems performance. We will then go through the results of the benchmarks and analysis. Finally in the last section we conclude about the performance of these OS's.

Hypothesis: Raspbian can outperform Pidora on the Raspberry Pi

II. RELATED WORKS

There is either little or no works comparing the two operating systems in this paper. There are many works on comparing other operating systems, computer performance measuring and benchmarking methodologies. This section will go over some of the works that influence the approach taken in this paper.

The three categories of an operating system are based on the breakdown in "Operating System Concepts" by Silberschatz [3]. This textbook does a great job of describing operating system processes and these categories should reflect main OS processes. The methodology of how to compare these operating systems and what benchmarks came from a couple of papers.

The first being an older paper from 1995 titled "Operating System Benchmarking in the Wake of Lmbench" by Brown and Seltzer. This paper still has one of the most in-depth looks at how to measure OS performance today. In this paper they describe how benchmarks are the only way to develop an understanding of operating systems and the computer hardware performance. They also explain how one needs a broad array of benchmarks that cover all the components of the operating systems [5]. In this paper we used multiple benchmarks in a variety of categories to follow this philosophy.

The way benchmarks were picked in this paper was based off of Chen and Lin's "A Systematic Methodology for OS Benchmark Characterization". Chen's paper was above how to categorize benchmarks. It pointed out that some benchmarks didn't even measure what it was supposed to. Such as disk benchmarks that didn't use large enough sizes to get around disk caching or CPU calculation using wrong timers [6]. This is why the benchmarks chosen in this paper we chosen carefully and all benchmarks are from the popular Phoronix Test Suite to verify popularity and correctness.

The methodology of how to pick and run the benchmarks was based off Martinovic and Balen's "Performance Evaluation of Recent Windows Operating Systems". This paper they compared three windows operating systems using a huge suite of benchmarks, which really encompassed the whole operating system. A subset of similar functioning benchmarks was used below, only difference is these could run on Linux and ARMv6 architecture. There benchmark methodology of running a benchmark and then reinstalling the OS to start fresh was borrowed also [7].

The way the secondary storage management section is benchmarked was influenced by Wanninen and Wang paper called "On Benchmarking Popular File Systems". In this paper the authors compared performance of multiple file systems on multiple operating systems. They used IOzone and their philosophy on file sizes and tests performed is used below [8].

III. OPERATING SYSTEMS

As stated in The Complete Reference: Linux, Linux is a fast and stable open source operating system for many devices. It has many features like Windows and OSX but what distinguishes Linux is its flexibility along with it being freely available [2]. This is where the Raspberry Pi comes in. Being that it has such specific hardware configuration there have been a handful of Linux distributions that are specifically developed with the RPi in mind. The two we will be comparing for the rest of this paper are Raspbian and Pidora.

Raspbian as stated earlier is a distribution of Linux specifically optimized for the RPi. More specifically Raspbian is a derivative of the very popular Linux distribution Debian. Debian has over 40,000 pre-complied packages and many equally popular distributions based on it like Ubuntu [10]. According to the Raspbian official website "Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware". It also currently offers over 35,000 pre-complied packages [9]. This makes Raspbian a very close port of Debian that is just tweaked slightly to suit the RPi's ARMv6 architecture. Raspbian is also based on the most current version of Debian, which is Debian 7.0 Wheezy. Also Raspbian is the most common operating system used on the RPi [1].

Pidora, like Raspbian, is a derivative of a popular Linux distribution but this time it is Fedora. Fedora currently has over 20,000 pre-complied packages [1]. Pidora is a Fedora Remix optimized for the Raspberry Pi and its ARMv6 architecture. Pidora is based of Fedora 18 [11]. Fedora 18 came out in early 2013. The current version is Fedora 20. Still Pidora has roughly 15,000 pre-compiled packages. These numbers still put Fedora/Pidora at the top of the charts for Linux distributions package number when excluding all the Debian based ones [1]. A note, package count does not mean anything when it comes to performance and is just stated here to outline the size and popularity of these OS's.

These two operating systems were chosen because they are both based on popular Linux distributions and are both well rounded operating systems that were specifically optimized for the RPi. There are a handful of other operating systems that run on the RPi such as Arch Linux, RISC OS, Slackware ARM and others. These were not included in the paper for a host of reasons, including time constants but are still good OS's and should be considered in future performance evaluation work.

IV. Methodology

An operating system does many things. As stated earlier a simple definition of an operating system is a program that manages the computer hardware [3]. More specifically operating systems manage hardware resources such as CPU time and memory space for user services like program execution or I/O operations. In general all of these operations can be broken down into three broad categories, process management, memory management and secondary storage management [3]. Most benchmarks are categorized based on hardware. These categories usually are CPU, memory, disk, graphics and others. In this paper we wanted to focus on benchmarking operating systems not hardware, even though they both influence each other. That is why instead we break down the benchmarks into three categories that more accurately represent operating system processes not computer hardware functions. Note that these categories are not independent of each other. They are interconnected. One cannot test secondary storage management without using process and memory management. The benchmarks will primarily test the category designated.

All the benchmarks that are used are from the Phoronix Test Suite, which is a comprehensive testing and benchmarking platform. It is designed to effectively carry out benchmarks in a clean, reproducible, and easy-to-use manner [12]. This test suite works on most Linux distributions including Raspbian and Pidora. Phoronix has over 100 benchmarks within it, and includes information on all of them. For this paper benchmarks were hand-picked to best evaluate the categories laid out.

A. Process Management

Process management is one of the most important things an operating system does. First off a process is a job, a unit of work within a computer system. A program in execution is a process. Processes need resources. A program does nothing unless it can be executed by the CPU. The operating system is responsible for managing these processes and the resources they use [3].

Since process management is involved is everything one does it can be tested in many ways. A very easy way to do this is to see how fast a system can calculate a problem. This should show how well the operating system can allocate and use CPU resources. An easy but computational intense problem is to calculate is finding prime numbers. The benchmark used to test this is called Sysbench CPU. Sysbench CPU is a basic CPU test that measures how fast (in seconds) a computer can calculate a user defined amount of prime numbers [12]. In our benchmark we used 20,000 prime numbers to stress the CPU out for a reasonable amount of time. The speed at which each of these operating systems can calculate this should be a good determination of the efficiency of floating point operations [7].

The next benchmark for process management is BYTE Unixbench. As stated on their website “The purpose of Unixbench is to provide a basic indicator of the performance of a Unix-like system” [13]. This benchmark is actually a suite of benchmarks, the one chosen is called Dhrystone 2. This specific benchmark focuses on string handling, no floating point operations like the previous benchmark SysBench. This benchmark is heavily influenced by compiler and linker options, along with integer data types [13]. The scores of this benchmark are in Loops per second (LPS), which is how many times it can cycle through a loop in the benchmark in a second. This will be a good evaluation of how quickly each OS can handle strings.

PHPbench is another interesting benchmark. This benchmark is a testing suite for PHP interpreter. This benchmark is CPU intensive. This benchmark outputs its results in terms of a score. This score is calculated by a ratio between the number of iterations and total time that was needed to perform all PHP tests. The number of iterations chosen was the default 1,000,000. Since both Operating Systems have PHP 5.4 installed this benchmark will be a good indication of how efficiently each can execute PHP programs and programs in general.

Another good way to benchmark process management is a Graphics test. This test would measure how well a system can utilize the GPU along with memory and secondary storage to render an image [7]. Unfortunately Pidora’s 3D acceleration graphics were not working at the time of this paper. Any graphics benchmark worthwhile need 3D acceleration so it is currently untestable. Even so both have both OS’s have GUI’s. Raspbian’s default desktop environment is LXDE. While Pidora’s is Xfce. Which desktop environment is “better” is based on preference but they both seem to leverage the available graphics hardware for responsive navigation during normal use. Actual graphics benchmarks will hopefully be possible in future work.

B. Memory Management

Memory Management is very important part of any operating system. The CPU directly reads and writes to the main memory during a process. The operating system is responsible for allocating and de-allocating memory space as needed. Since the CPU uses the memory directly it is very important that the operating system has the files needed in the memory or it will greatly reduce performance [12].

A good determinate of how well the memory management scheme works is to measure small read/write speeds. The benchmark I will be using is called RAMSpeed. RAMSpeed measures memory performance by allocating small amounts of memory space, then either writes or reads to it. It will do this with increasing amounts of data sizes until it reaches the memory boundary. This benchmark has two parameters, one being what unit to use and the other being how to record the bandwidth. For units integer and floating point were used (two separate runs) and the average bandwidth was recorded.

CacheBench is the next benchmark used to measure memory management performance. This benchmark is designed to evaluate the performance of the memory hierarchy by measuring bandwidth of repeated accesses to data items of varying vector lengths [14]. This benchmarks measures both main memory and cache performance. There are sub-tests for this benchmark and we decided to run three, Read, Write, and Read/Modify/Write. These sub-tests do what is implied by their names. These benchmarks together should show the memory management performance on each operating system.

C. Secondary Storage Management

Secondary storage management is another important part of operating systems. Because all data cannot fit into the small memory size it must be stored it on another storage device. In the Raspberry Pi's case it is an SD card. When a program is executed it must be loaded from the SD card to main memory. If there is a slow transfer of data this can limit the speed of the program substantially [7]. The operating system is specifically responsible for mapping, creating and deleting files for storage [3]. One way to test this is a lot like memory management tests, measure the speed of read/write performance. For this three different benchmarks were used, IOzone, AIO-Stress, and Unpack-Linux.

IOzone is a file system benchmark tool. This benchmark generates and measures a variety of file operations. We simply use the most basics file operation tests from this benchmark, write and read tests. The write test measures the performance of writing a new file to the system. This file was 1 GB in size making it bigger than the memory size on the RPI so disk storage was necessary. The benchmark times how long it take to write this file in MB/s. The read test measures the systems performance of reading an existing file. For this test a randomly generated file was read, again making it bigger than the memory to force disk reads. This test is also measured in MB/s. These two benchmarks should show the speeds of disk access which is very important to overall system performance [15].

AIO-Stress will measure the asynchronous input and output of a storage device. It does this by read and writing a 2 GB file sequentially multiple times. This should give a good evaluation of how well the secondary storage management scheme of each operating systems is when asynchronously reading and writing on the RPi.

Another simple but important test is to unpack the Linux Kernel and time how long it takes to do this. The Linux Kernel unpacked in this test was linux-2.6.34 which is 53.2 MB large. This test will time how long it takes, in seconds, to decompress this file. This is a good indication of how fast the secondary storage management scheme can handle moving and unpacking files which is a common operation in real world use [15].

All of these benchmarks metrics compared to each other with only the operating system changing should be a good indication of which one can utilize the Raspberry Pi's hardware the best and outperform in real world applications.

D. Benchmarking

For all the benchmarks the same process was followed. A fresh version of the Operating System was installed. The basic operating system was configured going through the setup wizard, and booting to terminal was enabled. The network was then setup to establish internet connection. Then all update/upgrades were installed using the included package manager. After the reboot phoronix-test-suite was installed and also the current benchmark. Then the benchmark was executed 10 times in a row and that data was recorded. After that the process was restarted with the other OS was installed. All tests are ran from the terminal.

Then comparing the results of these benchmarks there must be a verification that one group is in fact greater than the other by a statically significant margin or in other words, unlikely to happen due to chance. To establish this an independent, two-tailed T-Test was used. This is an inferential statistical test that determines whether there is a statistically significant difference between the means in two groups [16]. We can use this test to determine if an operating system was better in that specific benchmark. We then can evaluate which OS outperforms the other in each category. If Raspbian can perform better in all categories we can prove our hypothesis. All the detailed T-Test result are listed in the appendix along with full benchmark results.

E. Test Setup

Here is the technical information on the specific equipment used while running all of these benchmarks. The test setup consists of a Raspberry Pi, Wifi Dongle, power supply, hdmi connection to display, and sd card. The only thing that changes in the test setup is the Operating System installed on the sd card. The Raspberry Pi is a $35, credit card size microcomputer. The one in the test bench is a model b which had an ARM11 ARMv6 processor at 700MHz, a Broadcom VideoCore IV GPU at 250 Mhz and 512MB of memory shared with the GPU. The default memory split between the processor and GPU is 448MB to 64MB respectively. There is also a level 2 cache of 128 KB, which is only used by the GPU. The Wifi Dongle is a Ralink RT5370 802.11g/b/n 150 MBps Network Adapter. The power supply is a 1A / 5V micro USB. The hdmi cable is connect to a ViewSonic VX2452MH 24-inch 1080p monitor displaying at 720p. And finally the SD card is an 8 GB Sandisk class 4 SDHC card [17], [1].

When checking system resources from within each OS there were slight differences in what each system outputted as shown in Table 1. The most important differences are the Linux Kernels, memory, and disk filled. All of these could affect the performance of the system and will make the difference when it comes to the benchmark results.

Table 1**:** OS's Information

	Raspbian	**Pidora**
OS:	Debian Linux 7.2	Pidora 18
Kernel:	3.10.25t (armv61)	3.12.05.2013 (armv61)
Compiler	GCC 4.6	GCC 4.7.2
File System:	Ext4	Ext4
Screen Res:	1776x952	1794x954
Processor	ARMv6 @ .70GHZ(1 core)	ARMv6 @ .70GHZ(1 core)
Memory:	437 MB	435 MB
Disk Type:	8 GB SUO8G	8 GB SUO8G
Disk Filled:	2512 MB	1902 MB

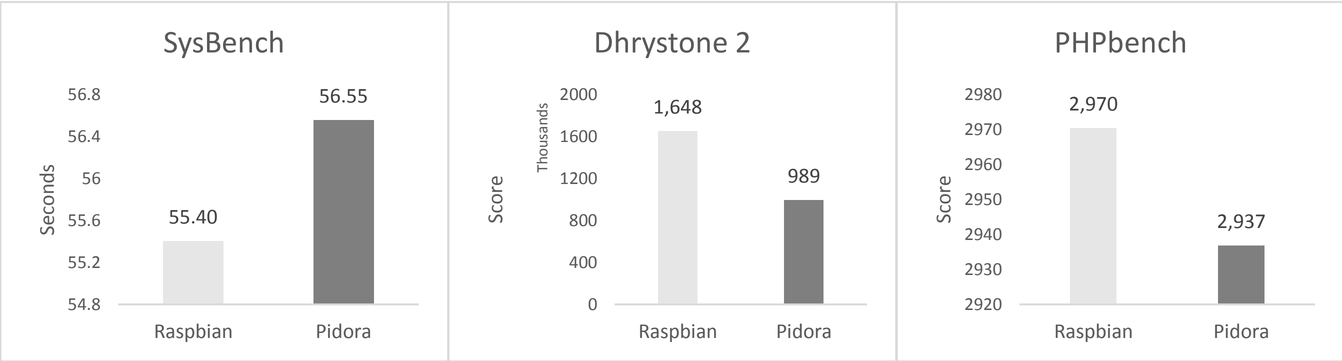


Fig. 3. Benchmark Results for Process Management

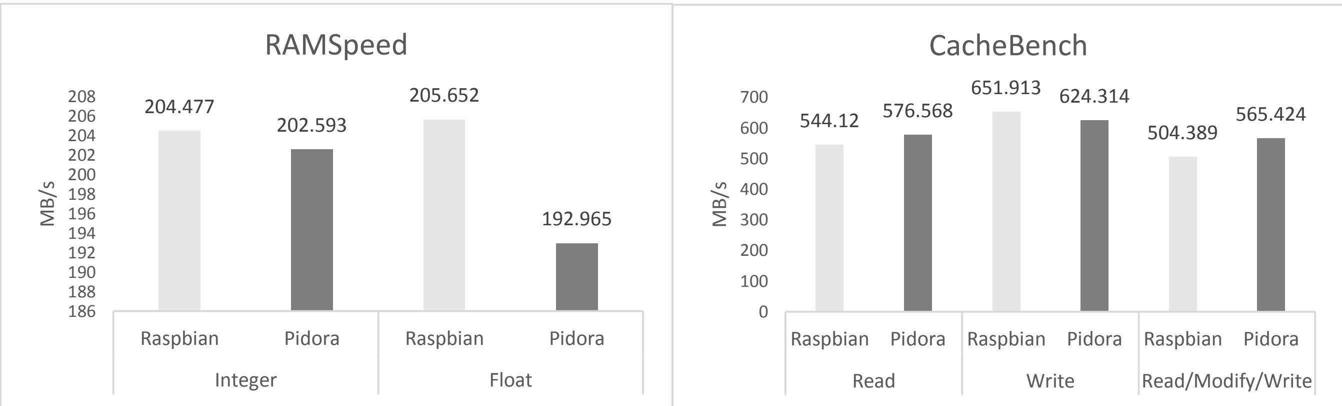


Fig. 4. Benchmark Results for Memory Management

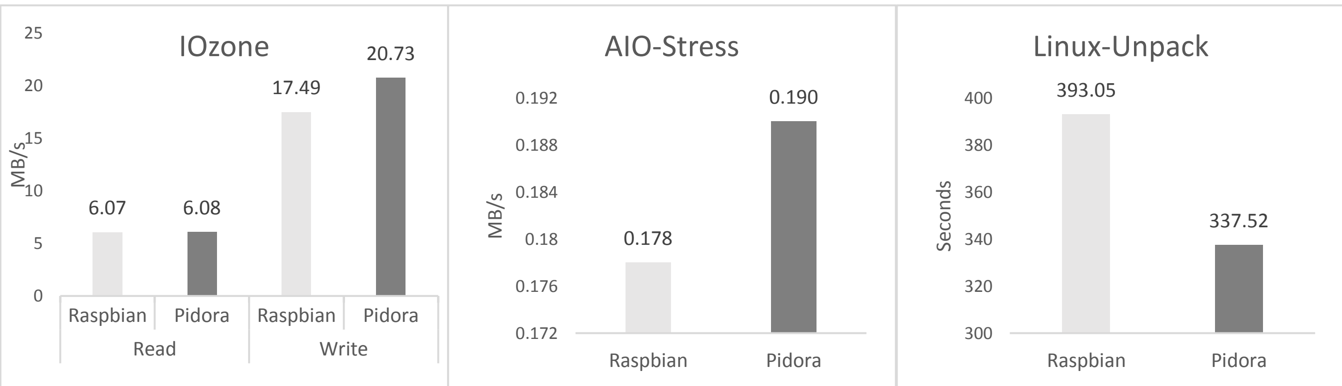


Fig. 5: Benchmark Results for Secondary Storage Management

V. Results

A. Process Management Results

As was stated earlier SysBench benchmarks measures how fast a computer can calculate an user defined amount of prime numbers. We decided to go with 20,000 prime numbers. As you can see in the left of Figure 2, the averages of these calculations are over a second apart. Being that less is better in this benchmark, we can say that Raspbian can calculate prime numbers faster than Pidora. This CPU and floating point intensive task implies that Raspbian can utilize this power than Pidora.

The next benchmark Unixbench's DhryStone 2 is a good indication of string handling in a system. In this benchmark Raspbian has a mean score of 1,647,782 and Pidora has a score of just 989,166. Being that greater is better in this benchmark Raspbian's score is 60% higher than Pidora's, as shown in the middle of Figure 2. This being a statistically significant margin Raspbian seems to outperform Pidora in string handling.

The last process management benchmark is PHPbench. This benchmark testing the PHP interpreter and is CPU intensive. As shown in the right of Figure 2, these benchmark scores and very close with Raspbian's average being 2970.3 and Pidora's being 2936.8. These are not statically significant from each other, meaning that neither operating systems comes ahead in this benchmark.

Process management is a very important part of every operating system. Raspbian outperformed Pidora in two of the benchmarks and the third was not statistically significant. This implies that Raspbian outperforms Pidora in the category of process management.

B. Memory Management Results

RAMSpeed is the benchmark that measures write and read performance of the main memory of the system using variable file sizes then takes the average. In this benchmark, as we can see in the right of Figure 3, Raspbian is about 2 MB/s faster on average on both memory speeds for integer. For Floating point Raspbian is over 10 MB/s faster. This is statically significant so we can claim that Raspbian is faster in this benchmark then Pidora.

CacheBench was out next benchmark. This benchmarked peak performance of very small memory calls. This benchmark was run with three different sub-tests Write, Read, and Read/Modify/Write and is shown in Figure 3. This benchmark shows Pidora outperforming in the Read and Read/Write/Modify. While Raspbian performing better in the Write benchmark.

Memory management is obviously very important in any system. In the benchmarks Raspbian outperformed in both RAMspeeds benchmarks and CacheBench's Write speeds. Pidora on the other hand outperformed in CacheBench's Read and Read/Modify/Write speeds. This means that we cannot definitively say that either of these operating systems outperformed the other in the category of memory management.

C. Secondary Storage Management Results

IOzone was a great test for measuring the performance of each operating systems secondary storage management. For the write test, as show in the left of Figure 4, Raspbian's write speed is 6.0673 MB/s versus Pidora's 6.0752 MB/s. Conducting a T-test on this data shows that these results are not statistically different. So write speed is a wash.

Now for the read test, shown in the left of Figure 4, one can see that Pidora has an average read speed of 20.7336 MB/s, while Raspbian is 17.4933 MB/s. Pidora has reads speeds over 3 MB/s faster than Raspbian's. With prove of a T-test this is statically significant and we can say the Pidora has faster read speeds than Raspbian.

The AIO-Stress benchmark measures the disk speed of multiple 2 GB random reads/writes. As the middle of Figure 4 shows, Pidora is about 1 MB/s faster than Raspbian when it comes to disk read/write speeds. This results is also statistically significant so we can claim the Pidora is faster than Raspbian for random read/write speeds to the disk by a slim margin.

The final benchmark was Unpacking a Linux Kernel. This benchmark measured how fast a system can unpack a Linux kernel in seconds, so less is better. As shown in Figure 4, Pidora can more quickly unpack the Linux Kernel then Raspbian by a statistically significant margin. Pidora can do this these on average 14% faster than Raspbian.

The secondary storage management is a very important part of an operating system. With these three benchmarks Pidora beats Raspbian in IOzone's Read, AIO-stress and Unpack-Linux speeds. With IOzone Write being inconclusive. Therefore we can state that Pidora can outperform Raspbian in the categories of secondary storage management.

VI. Conclusion

The goal of for this paper was to evaluate the performance of two operating systems, Raspbian and Pidora on the Raspberry Pi to determine if Raspbian utilizes the computer hardware better. To analyze this we took an approach of breaking down common functions of operating systems into three categories; process management, memory management, and secondary storage management. We then used synthetic benchmarks to simulate these categories when in use. We used the results of the benchmarks to compare the two operating system to determine which was better for a purely performance standpoint. In the end, Raspbian outperformed in one category, process management, while Pidora outperforming in secondary storage management and the third category was inconclusive. One operating system is not clearly outperform the other in all aspects, therefore the hypothesis of Raspbian outperforming Pidora is denied.

REFERENCES

[1] “About us” March 2014 RaspberryPi.org http://www.raspberrypi.org/about Accessed: 23 March 2014

[2] Petersen, R. (2008). *Linux: The complete reference* (6th ed.). New York: McGraw-Hill.

[3] Silberschatz, A., Galvin, P. B., & Gagne, G. (2005). *Operating system concepts* (7th ed.). Hoboken, NJ: J. Wiley & Sons.

[4] Balsa, Andre. August 1997. “Linux Benchmarking HOWTO” http://www.tldp.org/HOWTO/Benchmarking-HOWTO.html Accessed: 23 March 2014

[5] Brown, Seltzer. “Operating System Benchmarking in the Wake of Lmbench: A Case Study of the Performance of NetBSD on the Intel x86 Architecture” http://www.eecs.harvard.edu/ Accessed: 23 March 2014

[6] Chen, Chang, Lin, Yew. October 1-4, 2013. “A Systematic Methodology for OS Benchmark Characterization” http://delivery.acm.org Accessed: 23 March 2014

[7] Martinovic, Balen, Cukic. January 2012. “Performance Evaluation of Recent Windows Operating Systems” Journal of Universal Computer Science, vol. 18, no. 2. http://www.jucs.org/ Accessed: 23 March 2014

[8] Vanninen, Wang. 2009 “On Benchmarking Popular File Systems” http://citeseerx.ist.psu.edu/ Accessed: 23 March 2014

[9] “About Raspbian” 2013. Raspbian.org. http://www.raspbian.org/RaspbianAbout Accessed: 23 March 2014

[10] “WHAT is Debian?” Dec 6, 2013. Debian.org. https://www.debian.org/intro/about Accessed: 23 March 2014

[11] “Pidora” 23 May 2013. Pidora.ca. http://zenit.senecac.on.ca/wiki/index.php/Pidora Accessed: 23 March 2014

[12] “Open-Source Benchmarking” phoronix-test-suite.com http://www.phoronix-test-suite.com/ Accessed: 23 March 2014

[13] “Byte-Unix Benchmark” code.google.com https://code.google.com/p/byte-unixbench/ Accessed: April 2014

[14] ”Memory Benchmarks: CacheBench” asc.llnl.gov https://asc.llnl.gov/computing_resources/purple/archive/benchmarks/memory/membench_bm_readme.html Accessed: April 2014

[15] “IOzone Filesystem Benchmark” www.iozone.org http://www.iozone.org/ Accessed: April 2014

[16] Siegel, A. F., & Morgan, C. J. (. J. (1996). *Statistics and data analysis: An introduction* (2nd ed.). New York: J. Wiley.

[17] Maik, Schmidt. (2012). *Raspberry Pi: a quick start guide*. Dallas TX: The Pragmatic Bookshelf.

Evaluating the Efficiency and Straightforwardness of a Customized WKO Web-based Application

Matthew Genelin
Department of Computer Science
Winona State University
Winona, MN 55987
MGenelin08@winona.edu

Dr. Joan Francioni
Department of Computer Science
Winona State University
Winona, MN 55987
JFrancioni@winona.edu

***Abstract*—Whole Kids Outreach (WKO) is a non-for-profit organization in Missouri that provides low-income families with education and guidance for raising children. We designed a web application for WKO staff members to easily track and record client information. The interface of the web application was designed to be easy to use as well as efficient and straightforward. We conducted a formal usability study, consisting of ten participants, on this web application to simulate the major tasks of an Outreach Specialist staff person, which are: tracking and recording the results of a client visit and tracking the overall progress of clients. A post-test questionnaire was administered to ten participants as well as to five WKO staff members. The answers from the two groups were examined and compared. The usability test targeted learnability and robustness usability principles. Through analyzing the qualitative and quantitative results of the usability tests, we found that this application is efficient, straightforward, and user-friendly.**

Keywords—usability-study; user-interface; WKO; learnability; robustness;

I. Introduction

Whole Kids Outreach (WKO) is a non-for-profit organization in Missouri that provides low-income families with education and guidance for raising children. As of now, WKO currently is helping over 200 to 300 pregnant women and families with young children [8]. The staff at WKO consists of Outreach Specialists who are specially trained to treat and educate these families. The client data being recorded by the Outreach Specialists is confidential information and can be used if called upon in legal matters. The Outreach Specialists perform two main roles: tracking and recording results of a client visit, and tracking the overall progress of their clients over many visits. These specialists have been tracking their client's information using paper forms, which is very time consuming and a waste of resources. Using the WKO web application that we created the Outreach Specialists will be able to perform these tasks more efficiently and effectively.

Jeff Brookshaw, Brett Sissel, and myself, under the supervision of Dr. Francioni in accordance with Winona State University, have developed software for WKO. The software that was created for WKO was a web based application, using a customized user interface. The application connects to a database hosted on-site at the WKO center and consists of lengthy forms and detailed client information. These forms, as shown in Figure 1, are used to display information of an existing client by reading from the database. They are also used to submit any update on a new or existing client's information by posting to the database. The application was designed to be not only functional, but also user friendly and as straightforward as possible. The forms consist of a tabbed form view, and a user-friendly layout design. There is also a left navigation submenu, which allows easy access to any page in the application.

Since this is a new system, there is no baseline usability data on the web application. Although the application was designed with the user in mind, there is no way of knowing that it is easy to use and user friendly without testing it. A usability study can be used to provide data about the usability of the application as is now, and also provides baseline data of usability for future use. This way, if changes are to be made to the web application, the usability testing may be performed again to see where the modified web application then stands in comparison. A usability evaluation is an important part of the interface design process, which gives insight into what makes the interface a limited or successful interface [3].

A. *Usability Studies*

A usability study is a research methodology used to collect a user's tendencies and likings while in a controlled environment [1]. The principles tested in a usability study are learnability, flexibility, and robustness, as defined by Dix in [7]. Learnability is the ease with which new users can begin effective interaction and achieve maximal performance. Flexbility is the different number of ways in which a user and system exchange information. Robustness is the level of support provided to the user in determining successful achievement and assessment of goal-directed behavior [7]. A usability study consists of tasks/scenarios for the participants to complete. Each task performed by the user testing for one or more of these principles, where the results are analyzed to show the usability of a given application. A pre- and post-test questionnaire provides qualitative results about the users'

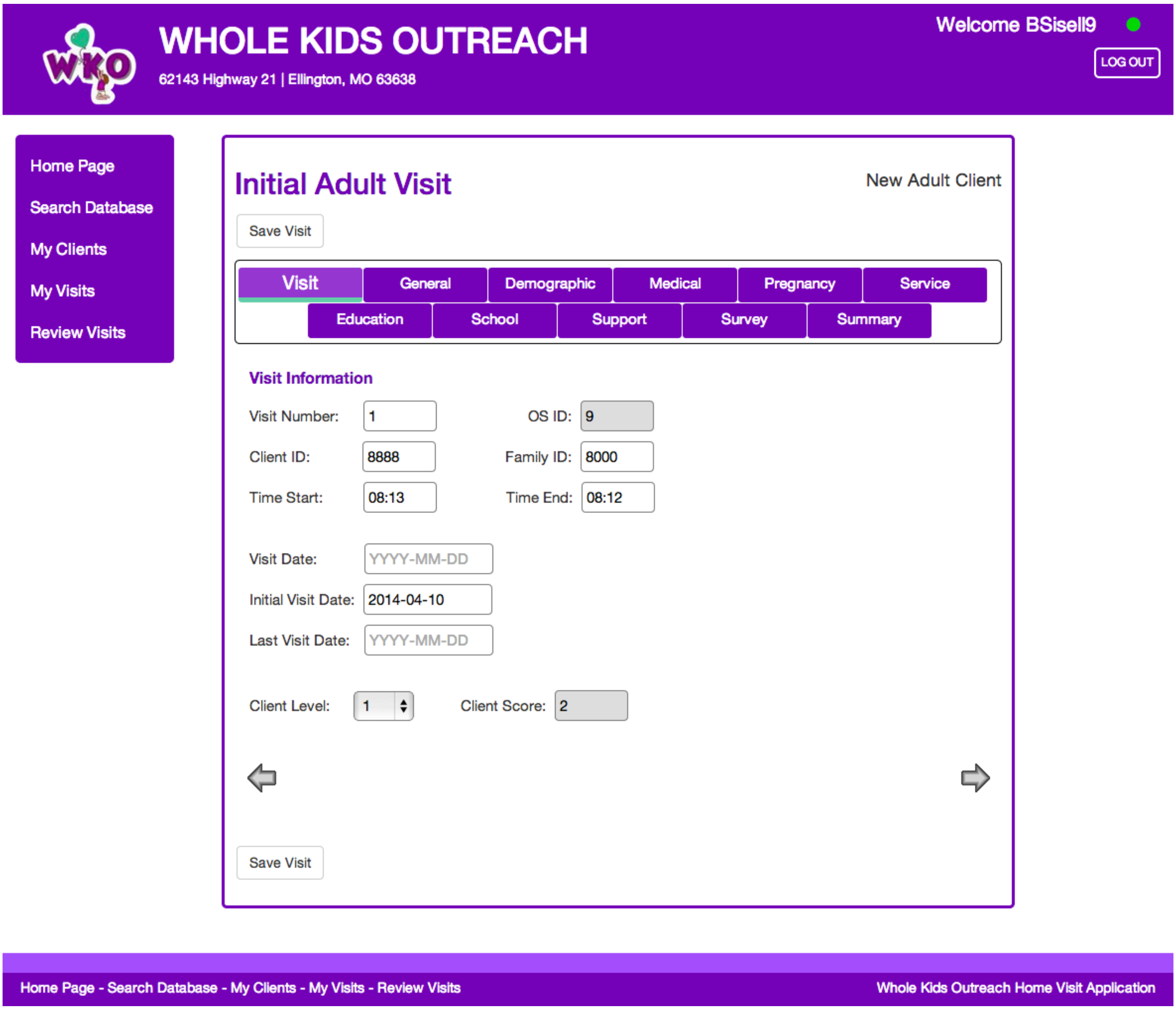


Fig. 1. Adult Visit Form within WKO application

opinion of the usability of a given application. The large amount of information recorded is qualitative, which gives insight about the design [4]. It is important for a web application or web site to be user friendly and easy to use. A usable web application makes for a more successful application. If a webpage or web application does not provide usability and accessibility then it will make it hard for the users to efficiently use the application [6].

This usability study focused on the learnability and robustness categories including the following specific principles: predictability, synthesizability, familiarity, and observability. Predictability is support for the user to determine a future action based on their past interaction history. Predictability was determined by how the participants were able to figure out a future correct path of action based on their understanding of what they have already encountered. Synthesizability is about the system support provided to a user to help them determine the effect of any prior operation on the current state. Synthesizability is related to the participant's understanding of what they did, and how they got to where they are. Familiarity is how the users' knowledge and experience within other real-world computer-based systems can be used when using a new system. This was related to the scenarios that included filling out the web forms, and how well they were able to begin using them. Observability is the extent of how the user is able to evaluate the internal state of the system from the representation on the user interface. Observability was used to test the participant's understanding of where they currently where, and where they could go [7].

Hypothesis: The assessment of a formal usability study on the Whole Kids Outreach Web Application will show that, with minimal training, the custom user interface is easy to use and more efficient than the current paper-based system.

II. Method

A formal usability study was conducted to provide usability data through analyzing the feedback and test data from a group of 10 participants. The software was set up for specific tasks to be performed by the participants. The following methodology tests the hypothesis that with minimal training the participants found the application easy to use and more efficient than a similar paper-based system. The usability study also gave useful information to the application developers regarding the design of the application, as well as anything the users liked or disliked, which may help improve the application. The following sections discuss the in depth creation, setup, and design of the usability study.

A. Participants

Ten participants from around the Winona area participated in the usability study. Among these participants, there were four females and six males between the ages of 19 - 58. There were also five WKO staff members who participated in the post-test questionnaire. The usability study was limited to only users who have used a Mac to eliminate any flawed data. Running a test of ten users gives statistically significant numbers and results [6]. The average Mac comfort level was a 3.6 out of 5, and none of the particpants have taken a usability study before. Of these participants two said to fill out forms less than once a month, one fills out one form a month, three fill out two forms a month, and four said to fill out three or more forms a month. After the tutorial was given, six particpants said that they felt they could use the application and four said that maybe they could use the application with more practice.

B. Setup

The software used was the WKO web application Beta version. It was run on a Mac Book Pro 12 with the operating system of OS X version 10.9. The Apache Web Server XAMPP 1.8.3-1 was running to process the PHP requests from the application.

C. Test Procedure

Each participant was first informed of what a usability study was, and then was read the test facilitator script. The facilitator script (see Appendix A) welcomed and thanked each participant for partaking in the usability study, and informed him or her of the purpose. They were reminded that the test is not a test of them, and that there is no right or wrong answer. Also, if for any reason they felt uncomfortable, they were able to exit the testing at any time. The participant was then asked to sign the consent and waiver form (see Appendix B) giving the permission to use their information for the study. After the participants signed the consent and waiver, they were given the tutorial and then asked the pre-questionnaire. The testing then started, and each participant was given a copy of the test scenarios. Each scenario was read aloud to the participants, where they were asked to complete each scenario. During the testing, it was important to stay calm and not make the participants feel as though they had done or were doing something wrong, which could have made the participant feel anxious, or add stress [2]. The participant was asked to inform us when they believed they were done, and then we would move to the next scenario. After the participants completed all of the tasks/scenarios, they were given the post-questionnaire and again thanked for participating in the study.

D. Training

Each user was briefed on what the study is about, and was told that no personal information was taken from them to ensure privacy. The tutorial consisted of informing the participant of the roles that each Outreach Specialist performs, including: the tracking and recording of the their client visit results, and the tracking of the overall progress of clients through their visits. There was also a 2-minute brief tutorial that familiarized the participants and showed the major functionalities of the application. This showed the participants the overall layout of the application as well as the layout of the forms

E. Pre- and Post- Questionnaire

A questionnaire was administered to each participant before and after completion of the study. The pre-questionnaire (see Appendix C) obtained general information of each participant including: their gender, age, a Likert scale of their comfort level using a Mac computer from 1 to 5 (1 being not very comfortable and 5 being very comfortable), how often the participant fills out online forms, whether or not the participant has partaken in a usability study before, and after the tutorial has been given if they believed they could use the application or not. The post-questionnaire (see Appendix D) obtained information of the users overall experience of the application. The information collected included: the overall impression of the application, a Likert scale of the ease of use from 1 to 5 (1 being hard to use and 5 being easy to use), what the participant liked best and least about the web application, what they would do to improve the app, a Likert scale of their comfort level of the application from 1 to 5 (1 being very frustrating and confusing and 5 being very straightforward and comfortable), and whether or not they believe the application is more efficient than a similar paper based system.

F. Test Scenarios

The usability tests were based on the two main functions that an Outreach Specialist performs, which include: tracking and recording the client visit results and tracking the overall progress of clients. To be able to observe participants, they should be given an assignment known as a task or scenario to complete [5]. Some of the tests/scenarios may be performed more often than the others, but together, they should cover the range of tasks an Outreach Specialist may have to perform. Test scenarios were developed for the following specific tasks:

1. Downloading an existing client's information (see Appendix E)
2. Starting a new visit on an existing client (see Appendix F)
3. Starting a new visit on a new client (see Appendix G)
4. Filling out a missed visit (see Appendix H)
5. Viewing the information on an existing client (see Appendix I)
6. Filling out a client termination (see Appendix J)
7. Deleting a client (see Appendix K)
8. Edit a previously recorded visit (see Appendix L)
9. Submitting a visit. (see Appendix M)

The test scenarios above follow a sequential order in which an Outreach specialist is likely to perform them. However, the test scenarios in this usability study were also made independent of each other, so that the completion of one task would not rely on the completion of a previous task. This ensured a participant could still perform a future task if they are unable to complete the current task. Table 1 shows the usability principles in each scenario tested.

An example test scenario was the deletion of a client. The test scenario was written as "An Adult client named Jennifer Smith has moved to a new address, which is outside of your assigned county. The address change has already been made, and another Outreach Specialist has already been given the client. You now no longer need this client to be downloaded on your laptop computer, and for security reasons want to delete the client. Can you delete Jennifer Smith from your list of clients?"

G. Pilot Testing

A pilot usability test was administered to two participants to assess the study as a whole. This pilot testing helped find problems with the wording of the test scenarios and questionnaires. The pilot testing also helped find the expected completion time and clicks for each task. These were determined by the expected completion time the designers determined before the sample testing, and the sample time and number of clicks the participants achieved during the pilot test.

H. Errors

User errors during the tests consisted of both non-critical and critical errors. Non-critical errors involved false and extraneous information entered. If the user were to misspell or add in extra information based on the given information, then they were deemed non-critical. These errors were considered non-critical errors because the participant was still able to find the location to enter in the data, but entered in the wrong information. Critical errors occurred when the participant failed to enter in the given data into the necessary field. These were deemed critical errors because the participant was unable to find the required field, which affected the usability of the application. Critical errors were recorded during the study, where non-critical errors were not.

I. Data Collection

The data was collected via two main methods. First a pre- and post-questionnaire collected basic information and feedback from all of the participants. The questionnaire focused mostly on the layout and design of the application, and anything in particular the participant liked or disliked. The second method involved the recording of the user's interactions with the computer and the results of the tasks. The participants were asked to perform a think aloud strategy while performing tasks and navigating through the app. The completion of the tasks was rated based on the success of the participant ranging from 0 to 2 (0 – not completed, 1 – completed with difficulty or help, and 2 – easily completed) and the number of clicks to complete the task. This method for collecting data allowed us to observe the interactions of the participant and the computer, as well as get an insight into what the participants were thinking as they went through certain tasks, and what made them click on certain objects.

TABLE 1. Test Scenarios with applied usability principles

	Predictability	Synthesizability	Familiarity	Observability
Scenario 1	X	✓	X	✓
Scenario 2	X	✓	✓	✓
Scenario 3	X	✓	✓	✓
Scenario 4	✓	✓	✓	✓
Scenario 5	✓	✓	X	✓
Scenario 6	✓	✓	✓	✓
Scenario 7	✓	✓	X	✓
Scenario 8	X	✓	X	✓
Scenario 9	✓	✓	X	✓

III. RESULTS

After the 10 participants completed the usability study, we analyzed the results and observations recorded from the tests and the answers to the questionnaire. The results were reviewed to find trends or multiple occurrences of situations and scenarios that may show the usability of the application.

A. Test Scenario Results

The test scenario results provided ample information to be analyzed. For each of the nine scenarios, the duration, number of clicks, errors, success rate and general comments were recorded. There was a predefined expected completion time and number of clicks defined for each scenario. Figure 2 shows the acutal and expected durations of each scenario in seconds, and Figure 3 shows the actual and expected number of clicks. (Figures on following pages) Note that if a scenario had a success rate of 0, then the time and click count was discarded. The actual values for time and number of clicks were computed as of each scenario for the group of participants.

As the participants moved through the scenarios, their times and number of clicks decreased in regards to the scenarios expected time and clicks. The first three scenarios each have a higher average completion time and number of clicks than the expected completion time and click count. The last six scenarios only have two scenarios where the average time and clicks are larger than the expected. The percent difference was calculated between the actual and expected times and number of clicks in regards to the first three scenarios, and the last six scenarios using a formula

$$\frac{|v_e - v_a|}{\frac{v_e + v_a}{2}} * 100$$

where v_e stands for expected value, and v_a stands for actual value. The first three scenarios had an average +15.14 percent difference in time and a +23.8 percent difference in clicks. The last six scenarios had an average -3.44 percent difference in time and -1.39 percent difference in clicks. This shows that the last six scenarios were completed on average faster and with fewer clicks than what was expected. The participants figured out the application relatively quickly, and were able to use the application efficiently.

The results clearly show that, scenario three had a significantly greater completion time and number of clicks than the other scenarios. For this scenario the user had to enter a relatively large amount of information into two different forms and save them. We noticed many of the participants went back over and reread the scenario multiple times. This is consistent with the higher error rate for scenario three when compared to the other scenarios. Since this scenario had the most information to be entered, the participant had the chance of making the most errors.

As stated earlier the success rate ranged from 0 through 2, where 0 was an incomplete scenario, 1 was a scenario that was completed with difficulty or help, and a 2 was an easily completed scenario. The average success rate and number of errors for each scenario are shown in Table 2.

Out of the 90 total scenarios completed by the participants, there was a total of 80 scores of 2, 9 scores of 1, and 1 score of a 0. This means that all of the scenarios were completed except for one, which results in a 98.9% completion rate. Out of all the scenarios, 88.9% were easily completed and 10% were completed with difficulty or help.

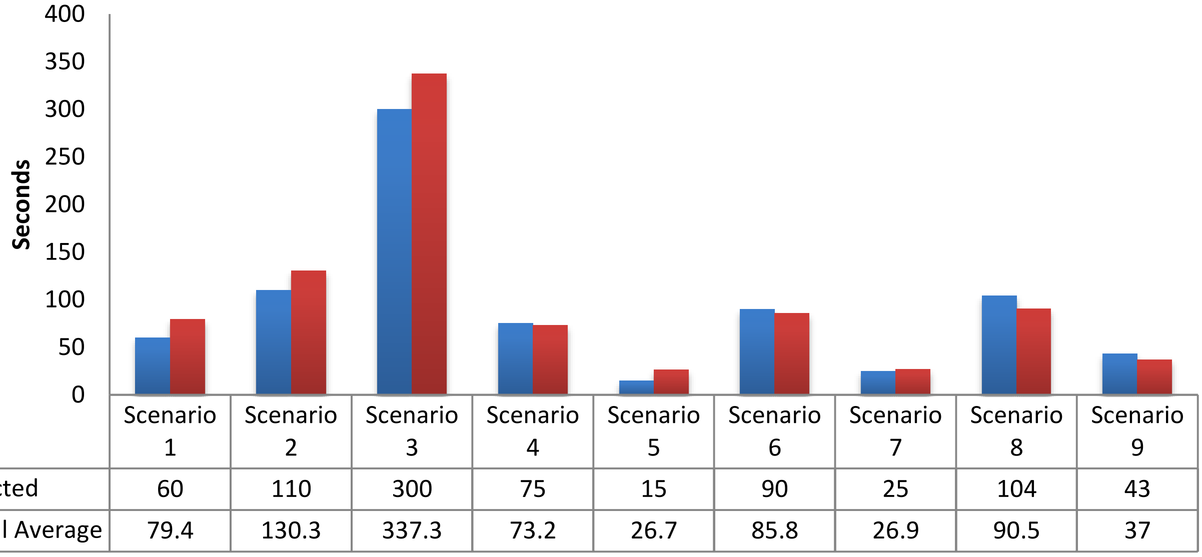
For the participants who took the study, the average Mac comfort level was a 3.6 out of 5, and none of the particpants had taken a usability study before. Two of the participants said they filled out forms less than once a month; one, one form a month; three, two forms a month; and four, three or more forms a month. After the tutorial was given, six particpants said that they felt they could use the application and four said that maybe they could use the application with more practice.

B. Post-Questionnaire Results

After the participants completed all of the scenarios, they were asked the post-questionnaire. Members of the WKO staff were also asked the same post-questionnaire to back up the results from the study. The main data taken from this survey to determine the usability was the participants' ease of use while using the application, the users experience using the application, and the efficiency compared to a similar paper-based system.

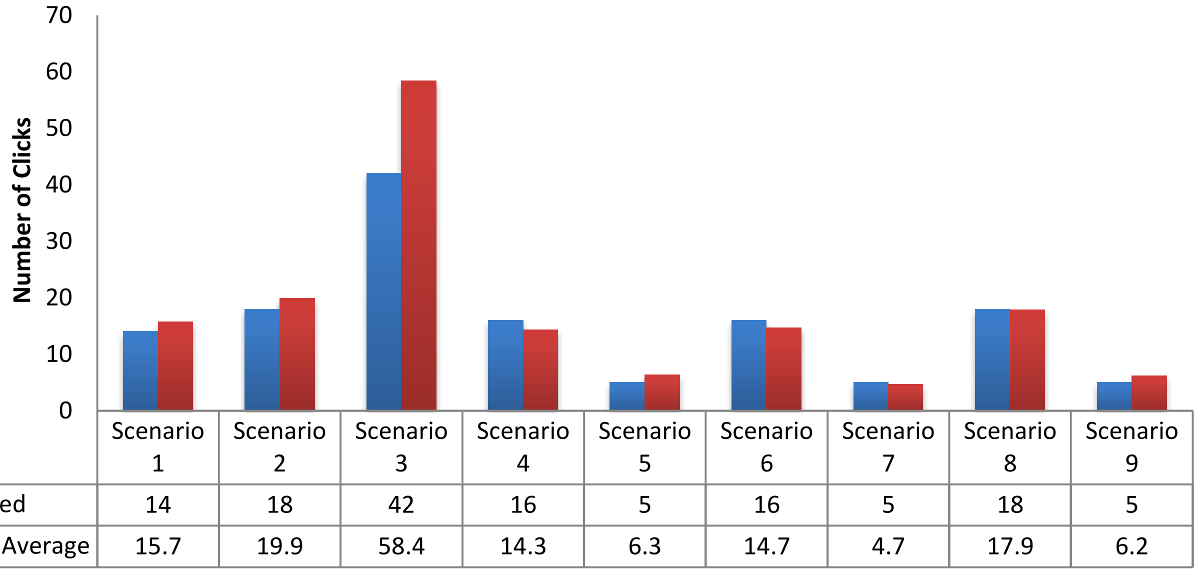
TABLE 2. Average Success Rate and Errors Per Scenario

	Success Rate (Avg.)	Errors (Avg.)
Scenario 1	1.7	0
Scenario 2	2	0
Scenario 3	1.7	1.8
Scenario 4	2	0.2
Scenario 5	1.9	0
Scenario 6	2	0.3
Scenario 7	2	0
Scenario 8	1.8	0.3
Scenario 9	1.8	0



	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Scenario 9
Expected	60	110	300	75	15	90	25	104	43
Actual Average	79.4	130.3	337.3	73.2	26.7	85.8	26.9	90.5	37

Fig. 2. Expected v. Actual Scenario Completion Times



	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario 7	Scenario 8	Scenario 9
Expected	14	18	42	16	5	16	5	18	5
Actual Average	15.7	19.9	58.4	14.3	6.3	14.7	4.7	17.9	6.2

Fig. 3. Expected v. Actual Scenario Number of Clicks

TABLE 3. Comparison of Participants and WKO Staff ease of use and experience of the application

	Participants from Study	WKO Members
Ease of Use	4.4	5
Experience	4.6	4.6

The participants the application's ease of use as a 4.4 and their experience using the application a 4.6, and the WKO members rated the ease of use a 5 and their experience a 4.6, shown as in Table 3.

The participants in the study and the WKO members both rated their experience using the application as a 4.6 on average. The WKO staff rated the ease of use as a 5 compared to the participants from the study's 4.4 rating. The WKO staff had been working with the software for approximately two to four weeks, where the participants were only given a two-minute tutorial on the application. Even so the participants still rated the application's ease of use close to the WKO members' rating. All of the participants found the application to be more efficient than a similar paper-based system. Many thought that it was more efficient for these main reasons: (1) the client's information backfilled for each visit so they would not have to retype it on each visit; (2) they liked being able to type in the information, and said it was faster than having to write out all of the information; and (3) they also thought it would be more organized.

IV. Analysis

A. Predictability

Predictability was determined by how the participants were able to figure out a future action based off what they had already encountered [7]. Predictability was tested in scenarios four, five, six, seven, and nine, which included filling out a missed visit form, viewing client data, filling out a client termination form, deleting a client, and the submission of visits. The average success rate for these scenarios was a 1.96. Three out of the five scenarios had an average completion time that was less than the expected completion time; scenario seven had an average completion time that was less than two seconds longer; and scenario five had a average time that was 11.7 seconds longer than expected. The scenario five had two ways to perform the scenario: view the client under the my clients page and view the client under the 'Searchdb' page. Five participants viewed the client under the 'My Clients' page and five viewed it under the 'Searchdb' page. Using the 'Searchdb' path usually took longer, and the five participants who performed the task this way had an average completion time 14.6 seconds longer than the group who viewed it under the 'My Clients' page. Each scenario that tested for predictability had an average number of clicks that was ± 2 of the expected number of clicks. Having a high success rate and low average completion time and number of clicks on the given scenarios shows the participants were able to figure out future actions based on what they had already encountered, and application supports good predictability.

B. Synthesizability

Synthesizability is the participant's understanding of what they did, and how they got to their current situation [7]. Synthesizability correlated with the completion of a scenario, and participants knowing the actions they took and the outcome of those actions. 89 of the 90 scenarios were completed, which is a 98% success completion rate. This shows that the participants were aware of the actions they were performing, and the results of those actions. Synthesizability correlates well with the success rate, time and number of clicks of the tests.

C. Familiarity

Familiarity is how the users' knowledge and experience within other real-world computer-based systems can be used when using a new system. Testing in this category focused on the scenarios that included filling out the web forms, and how well the users were able to begin using them [7]. Familiarity tested for scenarios two, three, four, and six, which were the scenarios the participant had to fill out client forms. Two of the four scenarios were completed with an average time less than the expected time, and all of the scenarios together had an average that was only 12.9 seconds longer than the expected time. Three out of the four scenarios had an average number of clicks that was ± 2 of the expected number of clicks, and scenario three had an average that was 16.4 clicks larger than the expected. This could be because it was a long scenario with a lot of information to be filled in, and two of the participants used a built in calendar option for the birthdates, which increased the number of clicks dramatically. The average 12.9 seconds longer than expected was not very long when completing these tasks.

D. Observability

Observability was the participant's understanding of where they currently where, and where they could go from that point [7]. Observability tested for the overall flow of the website, why they were on the current page, and what pages they could access at that point. The participants did not have any difficulty navigating through the application. Out of the 90 scenarios, only 1 was not completed. This shows the participants where aware of where they were, and where they could have gone. The biggest observation taken from the tasks was determining the difference between accessing the 'My Clients' and 'My Visits' page. Where the 'My Clients' page shows all existing clients and starting visit information, and the 'My Visits' page shows all current visits. In the post-questionnaire seven of the participants answered what they like best about the application with an answer dealing with the application's flow. The answers varied from the organization and layout to flow to easy to find and access items.

E. Improvements of Application

By tracking observations and participants' comments, we were able to determine if pieces of the application could be improved to increase usability. The most noticeable observation was that some participants had difficulty distinguishing the 'My Clients' and 'My Visits' pages functionalities. A participant noted that a 'help' button or subpage would be helpful in distinguishing between the functionalities of these as well as other pages. We also came across a smaller UI issue that could be changed to increase usability. When a user is asked to delete a client, a screen pops up to ask the user if they are sure and to type 'y' to delete. This pop up includes a long dialog sentence and a large text box to type in 'y'. A participant pointed out that the dialog could be shorter and to the point, and that the box could be shorter since they were only entering in 'y', or this functionality could be changed to a radio button with yes or no choices.

V. Conclusion

This usability study on the WKO application tested that with minimal training the application was easy to use and more efficient than the current paper-based system.

Although the testing went smooth, there were some changes that we would consider for future testing. The type of computer used would be judged with a higher concern. A Macbook Pro was used because that was the computer that was available. Although all of the participants had previously used a Mac computer before, many participants did not rate their comfort level very high. Also the possibility of allowing the participants the ability of using a mouse instead of the trackpad should be examined. Two participants had said during the testing that they prefer a mouse to a trackpad. Even though the Outreach Specialists run the application on their laptops, they are still able to use a mouse if they prefer. For a future study we could ask the participants before hand if they would prefer a mouse or to use the trackpad.

This study helped to provide baseline usability data, which is very helpful for any future changes made to the application. WKO may plan on adding in additional features to the application, or changes may be made to the existing application. Since the Outreach Specialists are traveling they may want to go to strictly mobile devices such as an iPad or other device. If any of these or other major changes are made to the WKO application, then this usability study may be duplicated to compare the usability data of the two applications.

Through testing the ten participants and comparing the questionnaires with the WKO staff members, we conclude that with minimal training the application is easy to use and more efficient than the current paper based system.

Acknowledgment

I would like to thank my project mentor Dr. Joan Francioni, as well as my three professors apart of CS 495 including: Dr. Mingrui Zhang, Dr. Sudharsan Iyengar, and Dr. Naryan Debnath. I would also like to thank all of the participants and staff from WKO that have participated in this study.

References

[1] Linfeng Li, Marko Helenius, and Eleni Berki. 2012. A usability test of whitelist and blacklist-based anti-phishing application. In *Proceeding of the 16th International Academic MindTrek Conference* (MindTrek '12). ACM, New York, NY, USA, 195-202.

[2] Rubin, Jeffrey, and Dana Chisnell. *Handbook of usability testing: howto plan, design, and conduct effective tests*. Wiley. com, 2008. Accessed Web. 22 Jan 2014

[3] Melody Y. Ivory and Marti A Hearst. 2001. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.* 33, 4 (December 2001), 470-516.

[4] Nielsen, Jakob. "How Many Test Users in a Usability Study?." *Nielsen Norman Group*. 4 Jun 2012. <http://www.nngroup.com/articles/how-many-test-users/>. Accessed Web. 22 Jan 2014

[5] McCloskey , Marieke. "Nielsen Norman Group." 12 Jan 2014. <http://www.nngroup.com/articles/task-scenarios-usability-testing/>. Accessed Web. 22 Jan 2014

[6] Richard Atterer. 2008. Model-based automatic usability validation: a tool concept for improving web-based UIs. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges* (NordiCHI '08). ACM, New York, NY, USA, 13-22.

[7] Dix, A. J., Finlay, J., Abowd, G., Beale, R. Principles to support usability, Human-Computer Interaction, 260-273, Third Edition

[8] "Whole Kids Outreach Programs." *Whole Kids Outreach*. <http://www.wholekidsoutreach.org/index.php?page=programs>. Accessed Web. 16 Feb. 2014.

On the Need for a General Language for General Intelligence

Theron Rabe and Sudharsan Iyengar
Computer Science Department
Winona State University
TRabe09@winona.edu

***Abstract*—General Intelligence can be assessed based on the accuracy, speed, and process with which one arrives at decisions. This is apparent in decision making processes where decision making is accomplished through deductive, inductive, and/or abductive reasoning. We first define the notion of a self evolving general language L , a superset of all languages. Additionally, we define and develop a process of reduction R, for improving the accuracy and speed of decision-making in L . Similar to natural languages, L is an incremental and self evolving language. Similar to intelligent processing, R accommodates all possible inputs. Finally, we present the limitations of λ-calculus with respect to L and propose remedies that provide us an implementation platform for L and R.**

Keywords—Artificial General Intelligence; General Language; Reduction; lambda calculus;

I. Introduction

The original goal of artificial general intelligence is to simulate intelligent behavior at or above human level [1]. This goal necessitates a means of comparison between intelligent agents. One such comparison method is to measure the speed, efficiency, and process with which the agent arrives at proper decisions [2]. In other words, when an agent is given some problem, its intelligence with respect to that problem is a combination of the amount of time it takes to find a solution, and how accurate that solution is. From this, we can argue that general intelligence should demonstrate speed and accuracy in solving not only pre-defined problems, but in general, an evolving scope of problems.

Humans use intelligence by reasoning over observations [2]. This reasoning falls under various forms of deductive, inductive, or abductive reasoning. If a person's repeated observations indicate that *A* must always imply *B* (i.e. *if A, then B* is learned), then upon subsequent observation of *A*, *definitely B* can be decided through deductive reasoning. Alternatively, if a person has at some time observed and learned that *A* possibly implies *B*, then upon subsequent observation of *A*, *maybe B* can be decided through inductive reasoning. Finally, if a person has learned through observation that *A* may imply *B*, then upon observing *B*, *maybe A* is decided using abductive reasoning. Since inductive and abductive reasoning are inherently uncertain, they are of particular interest for solving problems suited for general intelligence, which usually involve a high degree of uncertainty. Such problems include machine learning, pattern classification, natural language processing, statistical analysis, computer vision, and data compression [3,6].

Natural languages, the primary visible forum for intelligence, are incremental, self-evolving, self-mutating, and the evaluation of a statement using a language generates another statement in the same language. All decisions are based on previous experiences. Previously unknown expressions are either deemed irrelevant (and hence discarded) or considered acceptable new knowledge that is integrated into the language. Inductive and abductive reasoning requires earlier experience and decisions that help similar current decisions. For example, when a person (infers) induces that they will be hungry later because they did not eat breakfast, they must have previously experienced that not eating breakfast could cause hunger. The same can be said for a person who (feels) observes hunger and abductively attributes it to not having eaten breakfast. At the time of originally deciding that "not eating breakfast causes hunger", a minimum of two critical observations should have occurred. First, the person must have observed that they did not eat breakfast. Second, they must have later observed hunger. Any number of other intermediate observations (like "drank coffee" or "watched TV") may have occurred between these critical observations. Initially, all the relevant observations are included, considered, and processed in arriving at "not eating breakfast causes hunger". The number of observations included in such intermediate decision making can be called the critical observations' distance. Upon such repeated processing one tends to shorten the distance between observations in decision making. Thus, the act of correlating hunger to not eating breakfast is an act of shortening the distance between them (and pruning other insignificant observations).

When observations are later reasoned over, the observations that compose their distance may be ignored. For instance, if a person had "gone walking" after "not eating breakfast" and before becoming "hungry", the observations made on "gone walking" may not contribute to making the intelligent decision that "not eating breakfast" causes "hunger". The association of these two observations are utilized in inductive (forward or anticipatory) and abductive (backward or causal) reasoning.

This process of learning, and subsequent shortening of the distance between cause and effect observations, is elementary in demonstrative intelligence.

Thus general intelligence possesses a primitive decision-making process that shortens the distance between critical observations. This process lowers the complexity of the decision-making process by removing unnecessary intermediate steps. In other words, an intelligent agent must be able to take an observation series $A \rightarrow B \rightarrow C$, and simplify it to decide that observation A may directly indicate C without regard to the presence of observation B, where applicable. We call this decision-making process reduction.

Section II defines and describes the properties of the reduction process. Section III defines and describes the general language. Section IV presents the limitations of λ-calculus with respect to a general language and presents our approach to modifying λ-calculus so as to implement the features described in this paper.

II. REDUCTION - A WAY TO PROCESS PHRASES IN A LANGUAGE

Reduction is manifested at multiple layers of abstraction within intelligent thought. People use language as a means of abstracting their thought process. An observation, when abstracted by language, becomes a phrase. A phrase is either a symbol, or a sequence of symbols within a language. For example, "rain" is a phrase composed of a single symbol that represents the observation of water falling from the sky, in English. In the same way reduction permits simplification of non-critical observations, it permits simplification of their abstractions. When a complex phrase is interpreted by an intelligent agent, reduction can be applied to the phrase to shorten the relation between its sub-phrases or observations, thus simplifying the task of reasoning over their semantical correlations.

For example, let us take the sentence: "Enough humidity has gathered in the air as to generate clouds of an unmaintainable density" which could be interpreted to the phrase "It is raining".

This sentence has multiple subphrases (observations) viz. enough humidity, gathered in the air, generate clouds, and unmaintainable density.

Upon reasoning, the phrase becomes simpler but interprets the same. By utilizing a 'shortened' version of the original phrase, one is able to simplify the semantic interpretation of the original phrase. In other words, the reduced version is faster to interpret. With respect to language, reduction is the translation of phrases to semantically-equivalent (or -approximately equivalent), but syntactically-minimal previously learned abstract phrases.

We now present the properties of such a reduction process. Correlation between phrases and semantics, when indicated, are presumed. The establishment and verification of the semantics to phrases are beyond the scope of this paper.

Definition 1: A language $\mathcal{L}$ *is a tuple (T, N, G, S) where T is a set of terminal phrases, N is a set of non-terminal phrases, G is a grammar, and S is a semantics.*

Definition 2: Given a language $\mathcal{L}$*, a phrase P in* $\mathcal{L}$ *is a sequence of symbols of the form* $\{s_1, s_2, .., s_n\}$ *such that* $0<i<n,\ \forall s_i \in P\ (s_i \in (T \cup N))$*. All members of the power set* $\mathcal{P}\ (T \cup N)$ *meet the definition of a phrase.*

Definition 3: Given a language $\mathcal{L}$*, its grammar G is a set of production rules, each of the form* $A \rightarrow B$*, where A and B are phrases in* $\mathcal{L}$*.*

Definition 4:

$$N = \{A \mid (A \rightarrow B) \in G\}$$

$$T = \{t \mid (t \notin N) \wedge (\exists (A \rightarrow B) \in G : (t \subseteq B) \vee (t \subset A))\}$$

Definition 5: Given a language $\mathcal{L}$*, its semantics S is a set of tuples (t, b) where* $t \in T$*, and b is an observation. An observation is some mechanical or logical effect on an* $\mathcal{L}$ *interpreter.*

It is important to note that a phrase contains terminal and non-terminal symbols, but the semantics of the phrase is expressed by way of terminals only.

Definition 6: Given a phrase P, the set of symbols used in P is denoted $\{P\}$*. The distance of P is the cardinality of the set* $(P \cap N)$*, denoted* P_c*.*

Definition 7: Given a language $\mathcal{L}$*, the evaluation of a phrase P in* $\mathcal{L}$*, denoted* $P()$*, is a function such that:*

$$P() = \{b \mid \forall t \in (P \cap T) : (t, b) \in S\} \cup$$
$$\{P'() \mid \forall n \in (P \cap N) : (n \rightarrow P') \in G\}$$

Where P' is some partial evaluation of P.

An evaluation function correlates a phrase to its abstracted observations, thus causing a series of mechanical or logical effects on an interpreter. We argue that the evaluation of a phrase is dependent on the distance of the phrase. Terminals need no further reduction as they carry semantics.

Definition 8: The complexity of an evaluation, denoted $O(P())$*, is given as follows:*

$$O(P()) = 1 \qquad \text{if } \forall s \in P : (s \in T)$$
$$O(P()) = f(P_c) \qquad \text{if } \exists s \in P : (s \in N)$$

where f is some mathematical function

Definition 9: Given a language $\mathcal{L}$*, the reduction of a phrase P with respect to* $\mathcal{L}$*, denoted* $\mathcal{R}(P, \mathcal{L})$*, is a function such that* $\mathcal{R}(P, \mathcal{L}) = p$*, where p is a phrase in* $\mathcal{L}$*, and*

$$\mathcal{R}(P, \mathcal{L}) = \mathcal{R}(p, \mathcal{L})$$

$$P() = p()$$

$$O(p()) \leq O(P())$$

First, that the reduction of phrase P is equivalent to the reduction of its reduction, p. That is, the reduction function is final. Second, that the evaluation of the phrase P will be equivalent to the evaluation of its reduction, p. In other words, reduction does not change the semantics of a phrase. Third, the complexity of evaluating the reduced phase is less than or equal to that of the original.

An input string is reduced in formal languages by iteratively applying the rewrite rules specified in the language's formal grammar, on an input string, until it cannot be further reduced. Since natural languages have no exact formal grammar, their reduction is more difficult to achieve. Reduction of a natural language depends on an accumulated familiarity with the phrases that constitute the language. The correlations and equivalences amongst these accumulated phrases behave as the language's grammar. Because reduction of a natural language depends on phrases having been learned and subsequently used in an meaningful way, natural language reduction appears indicative of intelligence.

Thus, to replicate this act of intelligence using artificial systems, the reduction process must be achievable in a language that is being prescribed through free use of previously unknown phrases that could become part of the language. Thus our proposal for a framework for a general language as opposed to a specific natural language. Since general intelligence processes must be applicable in broad domains we define a general language next.

III. General Language

We note that the behavior of intelligence is dependent on what is known, understood, and utilized. Contrast this with an artificial system that can process phrases in the French language. This system is demonstratively limited in what it can accomplish because it is programmed as such, and it does not accommodate and/or learn other phrases. Humans on the other hand possess the ability to behave on what is assimilated, but additionally also accept and ingest new information, and thus evolve or grow. In fact, this is modus-operandi of human behavior. (Ironically, we consider this intelligent behavior and not the ability to process teraflops in milliseconds.) Importantly, note the language of a person is but that which has been assimilated and unrestricted, in contrast to what might be prescribed to be English, French, or the sign-language.

For the purposes of developing an intelligent machine we describe the notion of an unrestricted general language. This general language must satisfy the following three criteria:

- *General language must accept all possible phrases*
- *General language must be Turing-complete*
- *General language must be interpretable in-order*

Primarily, all potential phrases must be acceptable in the general language. This requirement implies that a general language has no predefined syntax rules. This is important as the order of the phrases is immaterial as long as the sentence is interpretable. Arguably, capability of interpretation without strict limitations on the order of the phrases, captures elementary intelligence. An example of this would be interpreting poetry as opposed to prose. Additionally, the general language must accept new previously unencountered phrases - as legitimate phrases. The interpretation of such phrases is subject to the intent of observations associated with the phrase and other considerations.

Secondly, the general language must have Turing-complete semantics, so as to enable inference of a type 0 grammar [7]. Given this feature, we can automate the grammar application of this language, giving us the possibility of developing an AGI system.

Thirdly, we note that intelligent behavior generally interprets observations as they are input - without the need for a pre-requisite forward (anticipatory) reference. As such, the general language must accomplish interpretation without a requirement of forward reference. This requirement is further explained.

Since, the general language lacks definite syntax rules, it must accommodate an infinite alphabet. An infinite set of symbols cannot be enumerated, as required for a formal grammar, but the set of contextually pertinent symbols can be. Consequently, during forward interpretation when a new symbol is encountered, the interpretation process must treat that symbol as a valid member of the language's alphabet in order to accept possible phrases with the new symbol.

Remedy 1: Represent infinite alphabet through its encountered subset.

This simplification permits an interpreter to reason a partial formal grammar over an alphabet. Note that as a consequence, the interpreter must posses the ability to maintain a dynamic alphabet and grammar rules. As a general language interpreter is used, it will encounter an increasingly large set of phrases. As such, it must maintain a repository of phrases encountered so far, and utilize this repository in its future interpretations.

Definition 10: A set of encountered phrases $\{p_0..p_n\}$, *represents an interpreter's history* $\boldsymbol{P}$.

Due to general language's need to be interpreted in-order, a function defined within phrase p_i must be expressed in terms relative to phrases $p_{0..i}$. In other words, the semantics of some future phrase is determined by its relation to past encountered phrases. Therefore, $\boldsymbol{P}$ represents a learned subset of the general language, as expressed in terms of $\boldsymbol{P}$. This makes $\boldsymbol{P}$ an evolving construct analogous to a human's understanding and use of natural language. For example, a person might equate the phrase “rain” to “water that falls from the sky”, but “water that falls from the sky” is just another phrase that can only be defined in terms of other learned phrases.

Definition 11: $\forall p_i \in \boldsymbol{P}\ (p_i() = f(p_0..p_i))$

where f is some computable function

Since a general language interpretation machine must be Turing-complete, it must support a means of defining and applying functions that support arbitrary recursion and abstraction. [4]

Definition 12: $\forall p_i \in \boldsymbol{P}\ \ (\exists A \in \boldsymbol{P} \wedge \exists B \in \boldsymbol{P} : p_i(A) = B)$

for any decidable $p_i(A)$

Therefore, a general language function is a means of rewriting arbitrary phrases into other arbitrary phrases, as derived exclusively from a set of encountered phrases. Because all formal grammars can be expressed as a set of phrase rewriting rules [7], all formal grammars can be directly derived from general language expressions. For this reason, deriving general recursive phrase→phrase rewrite functions by reasoning over $\boldsymbol{P}$ is equivalent to deriving a formal grammar for a language that contains all the same phrases as $\boldsymbol{P}$.

A machine that correctly interprets a general language, regardless of the semantics of that general language, will learn both the phrases and the grammar that constitute a subset of the general language. Since all languages are subsets of the general language, a general language interpreter can learn natural languages by interpreting an input that causes it to construct a $\boldsymbol{P}$ that is approximate to some desired natural language in both phrase content and grammar. Because reduction is a computable function for any language with a formal grammar and all computable functions may be contained in $\boldsymbol{P}$, approximation of a natural language via restriction of the general language permits reduction of that natural language with as much accuracy as permitted by the grammar defined in $\boldsymbol{P}$.

If semantics are defined for a general language, reduction of natural languages can be approximated. Reduction of a natural language is an act of intelligence that improves the speed and accuracy with which decisions can be made for problems with uncertain solutions.

We call for the need of a formal semantics for a general language. Given formal semantics for a general language, an abstract machine can be designed for evaluation of general language strings. A machine that evaluates general language has an inherent ability to learn, due to general language's requirement of an extensible alphabet. Furthermore, since the interpretation machine must be Turing-complete, it has the ability to derive and perform any computable function over its learned alphabet. Provided with the correct input string, an abstract machine that evaluates general language can learn both the phrases that constitute a natural language, as well as the functions that correlate those phrases within its language. Thus, a general language interpreter is capable of improving its intelligence with respect to any language, and therefore, any problem domain, through experience.

IV. A look at Λ-calculus and its limitations

To address the semantics for the general language, and exemplify the ambiguities that arise in doing so, we start with a Turing-complete language, and progressively remove all syntax rules. We use λ-calculus [5] as the starting language.

To exemplify the ambiguities that arise from removing syntax rules from λ-calculus, we will examine three syntactically invalid λ-expressions:

1. $\lambda xyz.a$
2. $\lambda\lambda x.F.a$
3. $\lambda\lambda x.xy.a$

Expressions (1), (2), and (3) each define a function whose body is composed of the symbol a and whose abstraction declaration contains syntax errors. Thus, in order for λ-calculus to meet the requirements of the general language, its semantics must be altered in such a way that each of these expressions is syntactically valid and unambiguously outputs the symbol a.

Expression (1)'s abstraction declaration contains three symbols (x, y, z) where only one is allowed by λ-calculus' formal grammar. To make this syntax valid, we suggest modifying λ-calculus such that a function with multiple symbols between λ and '.' is semantically equivalent to its fully curried version.

Remedy 2: $\lambda S.a = \lambda s_1.\lambda s_2.\ \dots\ \lambda s_{n-1}.\lambda s_n.a$

for any sequence S of symbols $s_1..s_n$

With this modification, Expression (1) becomes syntactically valid. And given any three inputs, Expression 1 retains unambiguous output of symbol a.

Expression (2) contains two consecutive λ symbols, so it can be referenced in parts. Call part "$\lambda x.F$ " the inner function, and everything else the outer function. Let F to be some oracle function that returns either symbol a or symbols xy. The output of F becomes the output of the inner function, which by way of Remedy (1) becomes the abstractions used by the outer function. Should F return symbol a, the outer function no longer outputs symbol a, and instead behaves as the identity function. Although the behavior of Expression (2) may arbitrarily change, it remains unambiguous in either definition it is dynamically given. We suggest the acceptance of semi-decidable function definitions by means of evaluating all definitions. Since definition is a prerequisite of application, any definition must be evaluated before its function can be applied. Because a function could potentially be applied immediately after definition, the expression containing its definition must be evaluated in-order.

Remedy 3: $\forall p_i \in P\ \ (p_i() = p_1(), p_2(), \dots, p_{i-2}(), p_{i-1}())$

Where f is some computable function, and p_i is a sub-phrase of phrase P

Strings must be evaluable in-order.

Expression (3) also appears to have an inner and outer function. Ambiguously, the inner function may consist of either $\lambda x.xy$ or $\lambda x.x$, depending on which function (inner or outer) owns symbol y. Should the inner function be provided another function for input x, that function x may be applied to one of two input sources, and in one of two orders. A function abstracted by x may be applied to y, or to whatever expression follows that which provided x. Additionally, that application may occur either before or after y has been provided with an expression to abstract. Which of these evaluation patterns is taken affects Expression 3's ability to output symbol *a*. To correct this ambiguity, we suggest marking both the start and end of both function definitions and function inputs with dedicated symbols.

Remedy 4: $\lambda x.y\ z \rightarrow (\lambda x.y)\ [z]$

By using these symbols purposefully and without restriction we can preserve the general language's first requirement (lack of syntax rules) and prevent ambiguity. This language is Turing complete and thus implementable on a computer.

V. Current Work & The Eesk Programming Language

We have designed and implemented a high-level programming language Eesk that attempts to be a general language. The Eesk system behaves as a lambda calculus interpreter that has, for the most part, remedied the ambiguities related to the double-lambda problem described above. With a few exceptions, this language meets all the three criteria of the general language.

The Eesk runtime environment has shown equivalent to an abstract machine that performs reduction on arbitrary learned languages for all halting inputs that have been tested. We intend to continue developing this system to use as a framework for further investigating the use of general language reduction as an approach to improving both the speed and accuracy of artificial general intelligence.

As with any correct implementation of the general language, Eesk's syntax is arbitrary. Valid Eesk is defined as any sequence of symbols. Conceptually, any symbol is either of the terminal or non-terminal type. Operators may be treated as terminal symbols. Operators that may be applied to an operand of one type may equally be applied to any operand of the other type. Thus, the language is weakly and dynamically typed. Since the typing is implicit, automatic, and prone to change, it does not necessarily concern an Eesk programmer.

Similar to other homoiconic functional languages like Scheme and Racket [8,9], Eesk is lexically scoped and full funarg [10] capable. The availability of symbols to their sub- and super-scope can be explicitly decided using "public" and "private" modifiers. Declaration of new symbols is done implicitly upon first encounter, defaulting to accessibility for all sub-scopes, but not the super-scope.

Due to general language's third requirement, Eesk may be parsed by a means as simple as LL(1) [11]. Each symbol encountered by such a naive left-to-right parser could be translated directly into machine code without respect to what symbols come next. The current implementation however, uses a recursive descent approach instead. Each descent may be implicitly escaped by encountering the end of a symbol stream. This solution permits much of the computational expense associated with determining scope to be handled at compile time.

To accommodate the remedies prescribed in this paper, Eesk employs a runtime architecture composed of three stacks, separating it from the list-processing approaches taken by philosophically similar languages [8, 9, 14]. The first of these stacks is used to store intermediate computed symbols, and the second to store function arguments. The Eesk calling convention causes these first two stacks to exchange responsibilities. This stack rotation method allows Eesk functions to both accept and produce syntactically arbitrary Eesk expressions without causing stack corruption. Furthermore, stack rotation permits the elements belonging to many sequential dynamic data structures to be accessed in constant time.

Eesk's third stack maintains control information for the calling convention, and its presence is opaque to an Eesk programmer. The third stack can be modeled using only the first two stacks, but in doing so, the runtime environment loses constant-time lookup of symbols in the super-scope.

Through the remedies provided in this paper, Eesk is a reflective language in which syntax is a first class citizen, and reduction of syntax is the primary mode of evaluation. Eesk expressions can be dynamically generated and evaluated by means of reduction. Beyond the primitive operators suggested for a pure reduction system, Eesk delivers additional predefined (but overridable) operator symbols that permit pattern matching between expressions, similar to use of (quote …) and (match …) in some languages [8,9] of LISP [14] heritage. Also, through intentional placement of function application operators, an Eesk programmer can explicitly denote whether a function is evaluated eagerly or lazily [12]. Additional features provided by the Eesk language framework include first class citizenship of continuations [13] and a foreign function interface.

VI. Conclusion

We have defined complementary tools of reduction and general language that characterize general intelligence in language processing. The process of reductions is aimed at simplifying the complexity of decision-making over uncertain problem domains. The beneficial and problematic implications of implementing such a framework is discussed. The use of λ-calculus, and suggestions for modifying its syntactic structure to make it suitable for use as the general language, are presented as well. We are calling on the need for the formulation of formal semantics of the general language as an approach to general intelligence.

VII. References

Allen Newell and Herbert A. Simon. 1976. Computer science as empirical inquiry: symbols and search. Commun. ACM 19, 3 (March 1976), 113-126. DOI=10.1145/360018.360022 http://doi.acm.org/10.1145/360018.360022

Sudharsan, Iyengar. Cognitive Primitives for Automated Learning, Frontiers in Artificial Intelligence and Applications, Vol. 171, AGI 2008, pp. 409-413.

Duda, Richard O., David G. Stork. Pattern Classification (Pt.1). (09 November 2000)

Turing, A. M. Computability and λ-Definability. The Journal of Symbolic Logic. Vol. 2, No. 4 (Dec., 1937), pp. 153-163

Church, Alonzo. An Unsolvable Problem of Elementary Number Theory

Murphy, Kevin P. Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series) (24 August 2012)

Chomsky, Noam. On Certain Formal Properties of Grammars. (1959)

Gerald Jay Sussman and Guy L. Steele, Jr.. 1998. Scheme: A Interpreter for Extended Lambda Calculus. Higher Order Symbol. Comput. 11, 4 (December 1998), 405-439. DOI=10.1023/A:1010035624696 http://dx.doi.org/10.1023/A:1010035624696

Matthew Flatt. 2012. Creating languages in Racket. Commun. ACM 55, 1 (January 2012), 48-56. DOI=10.1145/2063176.2063195 http://doi.acm.org/10.1145/2063176.2063195

[10] Joel Moses. 1970. The function of FUNCTION in LISP or why the FUNARG problem should be called the environment problem. SIGSAM Bull. 15 (July 1970), 13-27. DOI=10.1145/1093410.1093411 http://doi.acm.org/10.1145/1093410.1093411

D. J. Rosenkrantz and R. E. Stearns. 1969. Properties of deterministic top down grammars. In Proceedings of the first annual ACM symposium on Theory of computing (STOC '69). ACM, New York, NY, USA, 165-180. DOI=10.1145/800169.805431 http://doi.acm.org/10.1145/800169.805431

Paul Hudak. 1989. Conception, evolution, and application of functional programming languages. ACM Comput. Surv. 21, 3 (September 1989), 359-411. DOI=10.1145/72551.72554 http://doi.acm.org/10.1145/72551.72554

Reynolds, J. C. (1993). The discoveries of continuations. Lisp and symbolic computation, 6(3-4), 233-247.

McCarthy, John. Recursive functions of symbolic expressions. Springer Berlin Heidelberg, 1983.

ABXTester vs. Audio Exam: A Survey Comparing Audio Listening Test Apps

Aaron Sands

WSU Computer Science Department

Winona, MN

(651) 434-8619

ASands09@Winona.edu

***Abstract*— There are several data formats and bit rates available for digital audio listeners to use to suit their needs. Data formats each have their own characteristics. Lower bit rates offer smaller file size, and more efficient transfer over the Internet, but the result is generally poorer sound quality. The best way for a listener to compare differences between audio encodings is a double-blind listening test. Not much software is available to compare differences in sound fidelity, especially on Mac OS X. Participants were surveyed to determine if the new application Audio Exam provides a better experience than the existing application ABXTester in terms of intuitiveness, preference of test set-up, and the results generated by the applications. The results are summarized in this paper.**

I. INTRODUCTION

The Audio CD format became commercially available in 1982, and was the first time digital audio was available to consumers. As with any new technology, much of the public was skeptical at first, and many were stubborn to accept it [1]. It slowly gained popularity until it overtook the audio cassette as the most popular musical format around the early 1990's. Digital has many advantages over analog formats – higher fidelity, easier duplication, and more durability in the long term. In the early days of the CD, the personal computer was becoming more popular as well. By the late 1990's and early 2000's, personal computers started having enough secondary storage for a sizeable collection of music, and portable digital music players (MP3 players and iPods) started becoming immensely popular.

There are many ways to represent an analog waveform (music) as digital information that computers can process. Uncompressed digital audio, which is found on Audio CDs, requires large amounts of storage. Audio CDs contain PCM data, or pulse-code modulation. With PCM, an analog waveform is sampled at a regular interval, in the case of the audio CD, 44,100 times per second. This value is called sampling rate, and is usually expressed as 44.1 kHz. Each sample gets a fixed-size value. This is called bit depth, and in the case of the Audio CD, it is 16 bits. The bit rate, or number of bits per second needed to store one second of audio, can be found by multiplying sampling rate by bit depth by number of channels (nearly always 2 for music). The bit rate for PCM is 1,411.2 kbps. Some of the data occupied by a PCM file is unnecessary. For example, a period of silence is represented by a long string of zeros in the file. Compression codecs serve to eliminate some of this redundancy. There are lossless and lossy audio compression codecs. Lossless codecs decrease the file size, but allow the original waveform to be recreated exactly. FLAC, free lossless audio codec, is an example of a lossless codec. Lossy codecs such as MP3 reduce file size but also distort the waveform to some degree. Most lossy codecs allow control over bit rate at encoding time. Lower bit rates, in general, lead to more distortion and therefore poorer quality, potentially to the detriment of the listening experience. Depending on the bit rate, the result of a lossy encoding can be undetectable by a listener, in other words, transparent. Table 1 shows various encodings of a 3 minute, 57 second song, along with file size, and that number times 7160 (the average size of a music library) [2].

FORMAT	FILE SIZE	FILE SIZE * 7160
Uncompressed PCM (16-bit, 44.1kHz)	40.8 MB	292,320 MB (285 GB)
ALAC (Apple Lossless Audio Codec)	29.1 MB	208,356 MB (203 GB)
MP3 320kbps	9.1 MB	65,156 MB (63 GB)
MP3 VBR0 (220-260kbps)	8.3 MB	59,428 MB (58 GB)
MP3 256kbps	7.3 MB	52,268 MB (51 GB)
MP3 160kbps	4.6 MB	32,936 MB (32 GB)
MP3 128kbps	3.7 MB	26,492 MB (26 GB)
MP3 80kbps	2.3 MB	16,468 MB (16 GB)

TABLE 1. Various encodings of the track "Under Cover of Darkness" by The Strokes

Encoding a music library in a compressed format such as MP3 instead of CD Audio (uncompressed PCM) results in a lot of saved disk space. Audio listeners can benefit from knowing which bit rate offers the best trade-off of audio quality and file size. The ideal codec and bit rate depends on the listener's ability to perceive the difference in fidelity (which includes their own playback system and ears), how much music they own, and how much disk space they have. Judging what level of audio quality is good enough for the user is tricky, due to the nature of the human hearing system and psychological factors at work. The best way to tell is with a double-blind listening test of different encodings [3, 4].

ABX tests done with software are a common way to do a codec listening test. ABX testing involves comparing two stimuli "A" and "B" under a double-blind condition. For each trial, either "A" or "B" is randomly selected – this is "X". The test is to try to identify if "X" is "A" or "B". Double-blind means that neither the tester nor the person being tested knows which sample is which. In the case that a computer is administering the test, the random selection is done computationally. Over the course of many trials, statistical analysis is done on the results to determine the probability that the listener really can tell the difference, or that the subject was merely guessing. Many computer programs have been created to administer listening tests like this. They all serve the same purpose, but have different user interfaces and features. Program intuitiveness is very important in this context, as is the set up of the test itself, and the results generated by the program [3, 4, 5, 6].

Hypothesis: Users will find that the intuitiveness, ability to loop playback, and built-in results analysis in Audio Exam makes for a better experience than ABXTester.

II. METHODS

Only one piece of software previously existed in the Mac App Store that serves this purpose. It is called ABXTester, and is shown in Figure 1. In ABXTester, a user first loads the tracks 'A' and 'B'. Then, five randomized trials are displayed – each one is either 'A' or 'B', and the user can listen to as much of each file as they like with the audio players provided until they select 'A' or 'B' with the buttons to the right. Then, the answer is checked, and percent correct is shown. There is no way to loop one section of music or switch back and forth instantly – which is important for this type of test. The results must be recorded manually by the user and analyzed manually if they want to do so. In the newly created application Audio Exam, shown in Figure 2, a randomly selected 5-second section of the file is played, alternating between the two encodings. The file name, data format, and bit rate are shown onscreen as each track is being played. There are two possibilities – either the displayed file information is correct (a "true trial") or the information is swapped. (a "false trial"). At any time, the user can select 'true' or 'false' to complete a trial. During the test, the user has the choice to either quit or jump to another 5-second section. Once they pick which track they think is X, the program will record the result and jump to another point in the song. The reason for this test setup is to allow a very quick comparison between the two files without the user having to click pause and play several times. Upon quitting or reaching 20 trials, the program will display the number correct, the p-value of the cumulative binomial probability of the result, and the interpretation of the p-value. Although the setup of Audio Exam does not technically fit the definition of a ABX test, it is functionally equivalent to one.

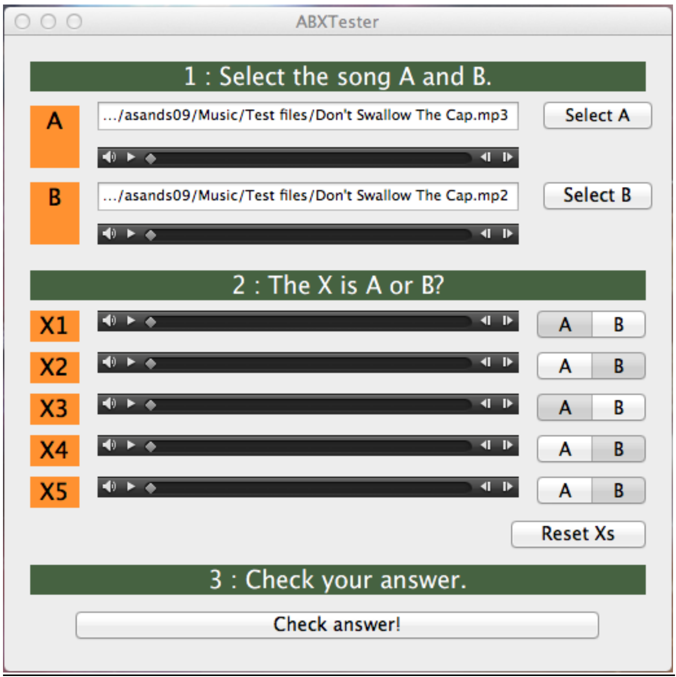


Fig 1. ABXTester UI

Fig 2. Audio Exam UI

The only material requirement the participants need is a Mac with Mac OS 10.9. The participants are emailed the audio files and instructions. The two applications can be downloaded for free in the App Store. Before beginning, the user downloads ABXTester and Audio Exam from the store. First, two test files are downloaded. Then, with each application, the respondents do ten trials comparing these two test files. The survey contains three pairs of questions which compare ABXTester to Audio Exam. The questions asked on the survey are as follows:

1A. Overall, ABXTester was intuitive to use.
1B. Overall, Audio Exam was intuitive to use.

2A. The way the each trial was set up in ABXTester was an effective and quick way to try to identify differences between A and B
2B. The way the each trial was set up in Audio Exam was an effective and quick way to try to identify differences between A and B.

3A. After the ten trials in ABXTester, I think I know how to interpret the result.
3B. After the ten trials in Audio Exam, I think I know how to interpret the result.

III. RESULTS

There were twelve survey participants. The median responses to questions 1A and 1B are *neutral* and *agree* respectively. The median responses to question 2A lies between *disagree* and *neutral*, and the median response to 2B is *strongly agree*. The median response to 3A is between *neutral* and *agree*, but the median response to 3B is *strongly agree*. The complete results are shown in the graphs below. Figure 3 shows the frequency of each type of response for questions 1A and 1B. ABXTester is in blue, and Audio Exam is in red. Figure 4 shows the responses for questions 2A and 2B, and figure 5 represents questions 3A and 3B.

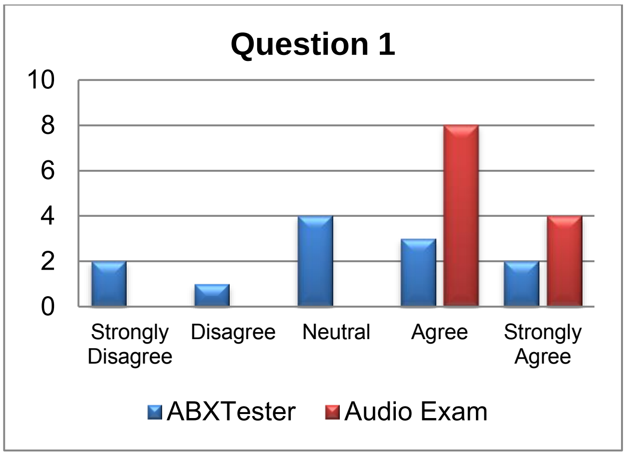


Fig 3. Number of responses to question 1

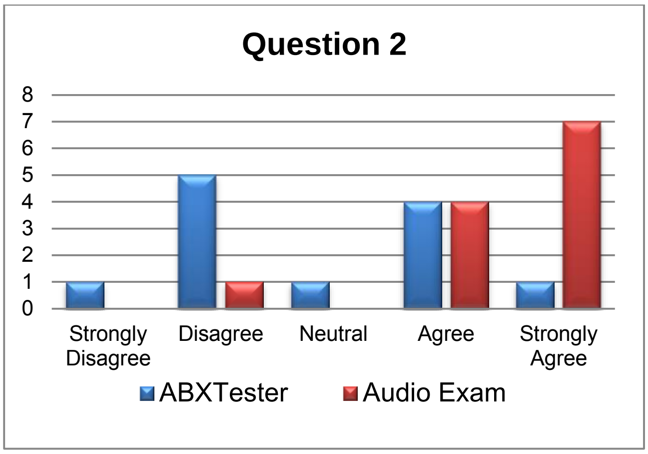


Fig 4. Number of responses to question 2

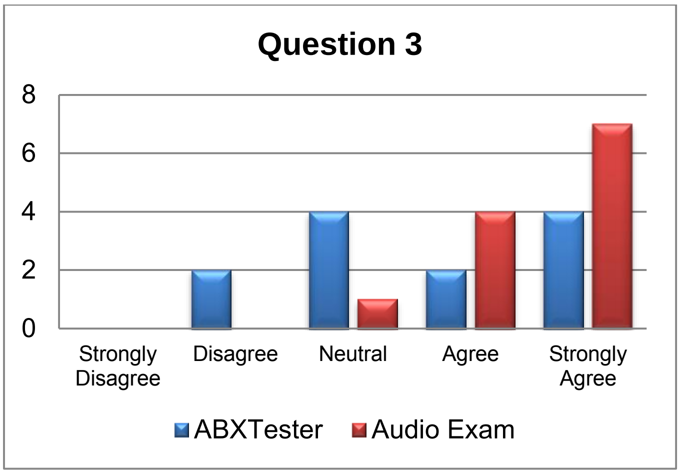


Fig 5: Number of responses to question 3

IV. ANALYSIS

The Wilcoxon Signed-Rank Test can be applied on each pair of questions (1A and 1B, 2A and 2B, 3A and 3B) to determine the likelihood that we would obtain these results given that users really didn't have a preference for either application [7]. The null hypothesis is that the median difference between each pair of data is zero. The alternative hypothesis is there is a nonzero difference in favor of Audio Exam. For question one, the test statistic (W) yielded is 55. Since the sample size excluding identical pairs is less than 10 for question 1, a z-score cannot be calculated, but since W is greater than the critical value of 29, the null hypothesis can be rejected. For question two $W = 66$, $z = 2.91$, and $p = 0.0018$ is obtained. Finally, for question three, the values $W = 55$, $z = 2.78$, and $p = 0.0027$ are found. Therefore, there is evidence that users preferred Audio Exam over ABXTester in all three of the aspects surveyed. From looking at the test statistics, the most dramatic difference was with preference of test setup (Q2).

There are some limitations to this survey. Firstly, twelve respondents is not many, and more data would have led to stronger confidence that one application is better than the other. As with any survey of this type, there is potential for central tendency bias (people tend to avoid the extreme answers), acquiescence bias (people tend to simply agree with the question stated), and social desirability bias (people may tend to choose the answer the asker wishes to see).

V. CONCLUSION

Although it seems Audio Exam is preferred over ABXTester, there is still room for improvement for Audio Exam. Some of the things that users suggested were improved file compatibility, drag-and-drop file selection, and a more detailed result output. The user interface in Audio Exam, as with any application, can be improved to provide an even more intuitive experience.

ACKNOWLEDGMENT

I express my sincere thanks to Dr. Minguri Zhang, Dr. Sudharsan Iyengar, and Dr. Narayan Debnath of the Winona State Computer Science department for their advising on this project. I also want to thank Dr. Tim Gegg-Harrison of the CS department for assisting with the software development and also Dr. Tisha Hooks from the statistics department for helping with the analysis part of the application. Thanks to Nate Clay and Josh Day for their help on the project as well.

REFERENCES

[1] "How the CD was developed." 2007. News.BBC.co.uk. http://news.bbc.co.uk/2/hi/technology/6950933.stm. Accessed: 6 Feb 2014.

[2] Sands, Aaron J. 2012. *Digital Sound: Where Music and Computers Come Together*

[3] "ABX Testing" 2004. Boston Audio Society.org. http://bostonaudiosociety.org/bas_speaker/abx_testing.htm Accessed: 6 Feb 2014.

[4] Meilgaard, Morten; Gail Vance Civille, B. Thomas Carr. 1999. Sensory evaluation techniques (3 ed.). CRC Press. pp. 68–70

[5] "ABX Binomial Probability Table" 2001. Home.provide.net/~djcarlst. http://home.provide.net/~djcarlst/abx_bino.htm. Accessed: 6 Feb 2014.

[6] Kleinholtz, L. L. 1992. Audio update. *Radio-Electronics*, 63(6), 76

[7] Boone, Harry N. *Analyzing Likert Data*. Journal of Extension

Wireless Helmet Sensor for Detecting Dangerous Impacts

Joel Sutton
Computer Science Department
Winona State University
Winona, MN, United States
Email: jdsutton7464@winona.edu

***Abstract*— Concussions in contact sports can go undetected due to a lack of observable symptoms. Concussion detection is essential for immediate treatment and monitoring as well as prevention of future concussions. It has been determined that an impact event with acceleration of 60*g* or greater can cause a concussion that may or may not display symptoms. We created a device to detect head impacts and send wireless alerts to a laptop for impacts that can cause a concussion. The system was created with an analog accelerometer, an Arduino microprocessor, and a simple XBee Wireless RF network. We tested the system by embedding the device in a football helmet and studying the consistency of the output generated by applying controlled impacts to five regions of the helmet. We found that while the system can reliably detect and transmit impact data, the consistency and usefulness of the data is limited by the use of a single accelerometer.**

Keywords—XBee, Arduino, Accelerometer, Helmet, Consussion, IEEE 802.15.4

I. Introduction

Concussions and the effects of head trauma are a growing concern in contact sports. According to Alcaraz et al. [1], 67,000 high school football players are clinically diagnosed with concussions each year, and nearly the same number of players suffer concussions that are not diagnosed. This provides two reasons for concern. The first is that a player who suffers brain damage and continues to play is at a higher risk for subsequent concussions. The second is that such injuries appear to be cumulative [2]. Bailes et al. [3] also suggests that head impacts that occur commonly during contact sports can result in subconcussive injuries. These injuries often have no outwardly visible signs and symptoms, so they are not recognized as concussions by a clinical diagnosis [4].

Diagnosis and treatment of players is based on physical signs and symptoms associated with a concussion. However, these studies suggest that clinical diagnosis is insufficient for detecting all head injuries and preventing subsequent injuries. Studies on the effects of controlled head impacts in laboratory animals and the effects of repetitive impacts in football indicate that an impact threshold in the range of 60-90*g* can cause a concussion [3]. Given that a significant number of concussions are not easily diagnosed, a wireless impact alert system may be a helpful tool for diagnosing and monitoring a player for concussive or subconcussive injuries. This project will create a prototype of a wireless impact alert system composed of a helmet module with an accelerometer, microprocessor, and an XBee transmitter that can notify a receiving module if an impact to the helmet exceeds a designated force threshold.

Hypothesis: A helmet sensor system composed of a 3-axis accelerometer, an Arduino microprocessor, and an XBee Wireless RF network can consistently detect helmet impacts from multiple directions and send alerts to a laptop.

II. Method

A. Materials

The hardware used to build this prototype system are an Arduino Uno, an ADXL377 3-Axis +- 200g accelerometer, two Digi XBee Series 1 Wireless RF modules with 802.15.4 firmware, an XBee shield, an XBee Explorer USB adapter, a 9V battery, and a MacBook Pro laptop. Other materials include a football helmet and a ballista impact device.

The Arduino Uno was chosen as the basis for this project for several reasons. Margolis [5] describes many advantages to using Arduino. The Arduino environment is designed for fast and effective prototyping. Arduino hardware and software (also referred to as Arduino) are open source and cross-platform and thus have a thriving and robust community of users and support that can be accessed easily online. Additionally, the Arduino hardware is on the same level of sophistication as the hardware used in commercial embedded systems. Various models of Arduino development boards are available, with sizes ranging from that of a credit card to that of a postage stamp. They come equipped with various I/O ports and can be powered with a battery power supply.

We chose the Arduino UNO, the basic model, because of availability of support resources and component adapters. These factors made it a good choice since we would be making frequent modifications and adaptations in our fast prototyping process. The disadvantage to the Arduino Uno for our particular application is that its dimensions are 6.8cm X 5.3cm, making it slightly larger than desirable for placing inside a football helmet. However, this project is intended to serve as a prototype, and a subsequent unit could be created using one of the smaller Arduino models.

We chose the Digi XBee Series 1 (802.15.4) OEM RF module as the wireless communication device, not only because of its small size, but also because of the flexibility provided by the IEEE 802.15.4 data transmission protocol. The ease with which the devices are configured is another attractive feature. The 802.15.4 standard allows for communication in both point-to-point and point-to-multipoint Personal Area Network topologies consisting of node devices and a coordinator device. In addition to these, the Digi XBee modules can operate in a peer-to-peer configuration, which does not require the use of a coordinator [6]. Configuration of new XBee devices into an existing network is simple and takes only minutes. The devices are capable of 250 kbps RF data rate communication to the end node, and this particular XBee model has a line of sight range of 100m [7]. These features are desirable since our device is to be used in fast-paced team sports.

B. *Hardware Assembly*

- The first assembly required was the connection of the XBee Shield to the Arduino microprocessor. This was done by following the XBee Assembly Guide provided by the XBee Shield manufacturer, Sparkfun [8]. The XBee Shield stacks on top of the Arduino. The pins of the XBee Shield attach directly to the header sockets on the Arduino.
- The next step is to connect one XBee Wireless RF module to the XBee Shield. To do this, the XBee Wireless RF Module was oriented according to the printed guidelines on the XBee Shield and the header pins of the module were inserted into the corresponding sockets on the shield.
- Connecting the accelerometer to the Arduino was done using solder and wire. Instructions were provided by the Adafruit Learning System [9]. Fig. 1 shows a diagram of the helmet device with labeled connections.
- The Arduino is powered by a 9V battery connected to the power jack on the Arduino board.
- The second XBee Wireless RF module is connected to the XBee Explorer USB adapter. Instructions for this are provided by the Sparkfun assembly guide [8]. The XBee Wireless RF Module is oriented according to the printed guidelines on the XBee Explorer USB adapter. The header pins of the module are inserted into the corresponding sockets on the adapter. The XBee Explorer USB adapter board is connected to the laptop by a miniUSB-to-USB cable.

C. *Software*

The XBee Wireless RF network was configured using the serial port terminal application called CoolTerm version 1.4.3. Configuration instructions are found on page 469 of Arduino Cookbook [6]. The MacBook Pro laptop was running OS X

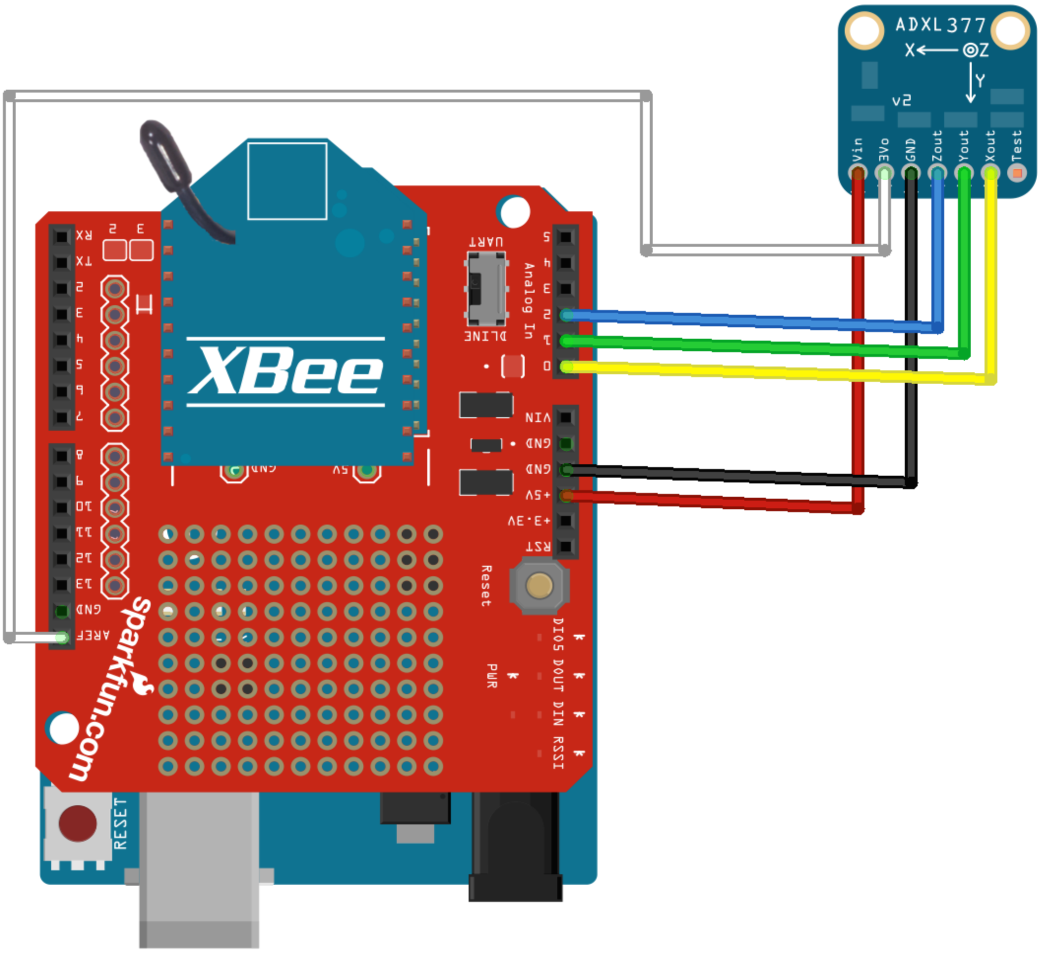


Fig. 1. Wiring diagram for connecting accelerometer sensor to Arduino UNO processor with XBee Shield and XBee Wireless RF Module

version 10.9.1 (Mavericks). The software running on the Arduino microprocessor was an Arduino sketch written in Arduino language, which is based on C and C++. The coding was done using the Arduino IDE version 1.0.5.

D. Data Processing and Output

The accelerometer senses acceleration from force applied to the accelerometer on the X, Y, and Z directional planes and outputs a corresponding voltage value to the Arduino A0, A1, and A2 pins respectively. The Arduino sketch reads the input values of the A0, A1, and A2 pins in a continuous loop.

Since our accelerometer is connected to the Arduino 3V regulator pin, the accelerometer voltage output when level and at rest is 3V. The Arduino has a 10-bit Analog to Digital Converter, or ADC, that converts the analog voltage values from the ADXL377 into digital values ranging from 0 to 1023. For a resting, level accelerometer, the ADC value is approximately 512. In order for our application to make use of this data, our Arduino sketch converts the ADC values corresponding to the X, Y, and Z values from the accelerometer into acceleration g values. The conversion function is derived from the Arduino ADC conversion function and the ADXL377 Zero g Voltage offset factor and Sensitivity Ratiometric factor as listed on the ADXL377 Data Sheet [10]. The sequence of events for processing I/O is as follows:

- A helmet collision event occurs and the accelerometer senses the impact.
- The accelerometer outputs analog voltage values for the X, Y, and Z-axes to the Arduino processor. The ADC converts each analog value to a digital value ranging from 0 to 1023.
- The Arduino sketch converts the digital value into an acceleration g value for each axis.
- The values are composed into a single string message with axis labels corresponding to the value for that axis
- If the acceleration g value is $\geq$ 60 g, the string "DANGEROUS HIT" is appended to the message.
- The resulting string is written to the XBee Wireless RF module via the Arduino serial output.
- The XBee Wireless RF module uses the 802.15.4 wireless protocol to send the message to the XBee Wireless RF module connected to the laptop.
- The receiving XBee module writes the received message to the USB serial port on which the CoolTerm application is listening.
- The CoolTerm application displays the message in the console for a human observer to read.

E. Helmet and Helmet Sensor Unit

The helmet used in this experiment is a size X-Large Rawlings Momentum youth football helmet. The helmet was purchased new and was unused prior to this experiment. The impact sensing unit was affixed to the inside of the helmet with the accelerometer placed at the top-center of the helmet interior and connected to the main unit by jumper wires. The accelerometer was oriented such that the X-axis sensed impacts on the front and rear of the helmet, the Y-axis sensed impacts to the side of the helmet, and the Z-axis sensed impacts to the top of the helmet. The main unit, consisting of the Arduino, X-bee module, and battery pack, was attached to the helmet interior with adhesive tape and additional foam padding.

F. Impact Device

The impact device is a ballista style machine that uses an elastic sling to propel a blunt-ended bolt into the helmet. The ballista has a base platform with two vertical stanchions on one end. On the opposite end is a platform where the helmet sits. The distance from the top center of the helmet to the closest stanchion, the front stanchion, is 1m. Each end of the elastic sling is anchored on either side of the front stanchion. The elastic sling is made of rubber tubing with a 13mm outside diameter. The bolt shaft is a 1.3m length of PVC pipe with 26mm outside diameter. The bolt head is made from lead fishing weights that are bundled, wrapped, and affixed to the PVC shaft with duct tape. The total weight of the head is approximately 226 grams.

To operate the device, the bolt is inserted into guide holes in the stanchions with the blunt end toward the helmet platform. The bolt is nocked on the elastic band and pulled back to the desired draw length. Markings on the bolt shaft are used to consistently measure draw length. The zero marking is 38cm from the nock end of the shaft and indicates the point on the shaft that rests at the front stanchion when the bolt is nocked and the elastic is drawn to the point just before tension. From the zero mark, the shaft is marked at 10cm increments up to 50cm, which is the maximum draw length we used for firing at the helmet. When released, the bolt is launched at the helmet sitting at the opposite end of the platform. Fig. 2 shows the ballista device with the helmet on the platform in preparation for an impact test on the facemask region of the helmet.

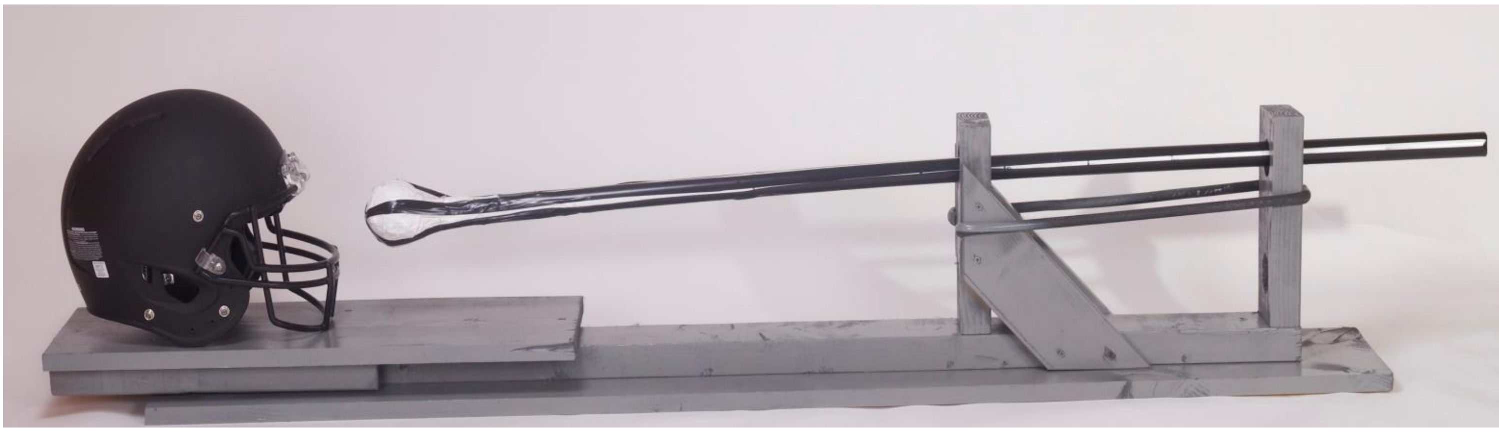


Fig. 2. Ballista impact device with helmet in place for facemask impact test

G. Testing Procedure

The system was tested by performing a series of controlled impact tests on the helmet and recording the output of each impact as it was displayed in the CoolTerm console on the laptop. Data was collected for impacts on 5 helmet regions: facemask, side, forehead, front, and back. These regions are the same regions used for impact data collection in the study by [4]. Each helmet region was subjected to sets of 5 impacts from the 20cm, 30cm, 40cm, and 50cm ballista draw lengths so data could be recorded as the impact force increased incrementally. This makes a total of twenty recorded impacts for each helmet region. For each impact, the primary impact data for the X, Y, and Z-axes was captured in the CoolTerm console and recorded in a spreadsheet. Fig. 3 shows a sample output from a trial run of the system.

In theory, the device would only transmit impact data for impact events that are considered to be dangerous, i.e., impact events with force ≥60*g*. However, because we wanted to examine the consistency of the data over a range of impact forces, we had to modify the Arduino code to also transmit data that would not necessarily indicate a dangerous impact. For our experiment, the threshold for a sensed impact force to be transmitted was 4*g*. This means that any impact to the helmet that was sensed as less than 4*g* would not be transmitted to the laptop unit. If, for any given hit event, no data was transmitted for a particular axis, a value of <4 was recorded for that axis in the spreadsheet. In order to simulate the functionality of alerting the laptop observer of actual dangerous impacts, we programmed the Arduino to append a 'DANGEROUS HIT' message to the output when an impact ≥60*g* was sensed. Our spreadsheet record also indicates whether or not this message appeared as expected for dangerous impact values.

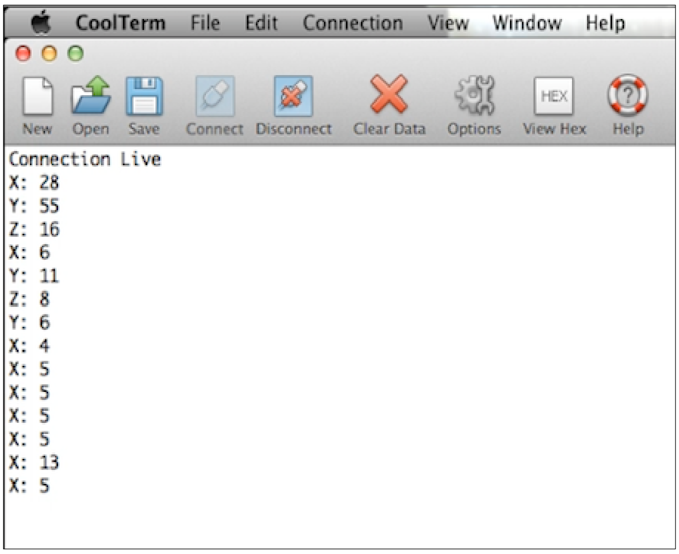


Fig. 3. CoolTerm console displaying output of side impact test

Lowering the impact threshold for data transmission introduced the side effect of transmitting data that would be considered 'noise'. While the noise is legitimate impact data, it does not reflect the primary impact of the bolt on the helmet. Instead, it is impact data sensed when the helmet landed and rolled in the catch padding after being knocked from its platform. The highest values for X, Y, and Z are shown clustered together. These are the initial output corresponding to the primary impact of the bolt on the helmet. They are indicative of the data points collected as our result data for each trial. The remaining data was discarded as noise.

III. Results & Analysis

Results were recorded for 100 impact trials. Each result contains 5 data elements:

- Draw Length – indicating the draw length of the ballista in cm.
- X - indicating the impact value for the X-axis.
- Y - indicating the impact value for the Y-axis.
- Z - indicating the impact value for the Z-axis.
- ALERT - indicating YES if the output contained the 'DANGEROUS HIT' message for any X, Y, or Z value ≥ 60*g*
- Note: An X, Y, or Z column label denoted with * indicates that this axis is the primary axis of impact for that helmet surface region. An example is shown in Table 1, which displays data for the facemask helmet region.

For each impact test, we recorded the X, Y, and Z-axis outputs for a total of 300 data points. Analysis focused on the 100 data points corresponding to the primary axis receiving the impact for each helmet region: X-axis for the forehead, facemask, and back, Z-axis for the top, Y-axis for the side.

To simplify the comparison of data for each region while accounting for the wide range of output values within each helmet region, we calculated the arithmetic mean of the 5 output values produced by each set of draw length tests on each helmet region. The mean values for each helmet region are listed in Table 2. This gives us one value for each helmet region that can be compared to the mean impact value of the other helmet regions for the set of tests performed at each draw length.

This data shows that the mean values for each helmet region vary significantly. For the 20cm draw length tests, the back helmet region had the lowest mean impact value at 25*g*, whereas the highest mean impact value was for the side helmet region at 54*g*. This is a difference of 29*g*. The largest range of mean impact values among the helmet regions was produced at the 40cm draw length. The lowest mean value produced by this set of tests was from the top helmet region at 67*g*. The highest mean value for this set of tests was 135*g* produced by impacts on the side helmet region. This is a difference of 67*g*. Mean impact values for the helmet regions follow a similar pattern for the 30cm and 50cm draw length test sets.

TABLE I. FACEMASK REGION TEST RESULTS WITH PRIMARY IMPACT AXIS X

DRAW LENGTH(cm)	X* AXIS(g)	Y AXIS(g)	Z AXIS(g)	ALERT
20	37	18	28	
	54	18	5	
	39	14	6	
	55	5	<4	
	37	14	19	
30	94	14	5	YES
	41	14	5	
	63	7	8	YES
	42	5	13	
	83	31	12	YES
40	82	31	12	YES
	64	21	24	YES
	99	14	<4	YES
	40	13	47	
	85	33	20	YES
50	116	47	12	YES
	16	63	23	YES
	21	5	21	
	67	38	28	YES
	32	47	68	YES

TABLE II. MEAN OUTPUT OF FIVE HELMET REGIONS FOR EACH DRAW LENGTH

DRAW LENGTH (cm)	MEAN SIDE IMPACT (g)	MEAN FACEMASK IMPACT (g)	MEAN FOREHEAD IMPACT (g)	MEAN BACK IMPACT (g)	MEAN TOP IMPACT (g)
20	54	44	27	25	42
30	58	65	35	38	55
40	134	74	72	94	67
50	108	50	88	82	91

Despite the inconsistency in the data within and among each helmet region, one trend is common among the helmet impact regions. The trend is noted when comparing the change in the mean values for each helmet region as the draw length of the impact tests increases. As we might expect, an increase in the draw length of the impact test corresponds to an overall increase in the mean impact values for each helmet region. For instance, the mean impact value for the side helmet region is 54*g* for 20cm draw length tests. It increases to 108*g* for the 50cm draw length tests, reflecting an overall upward trend. However, it is interesting to note that the mean impact values of four of the five helmet regions is highest for the 40cm draw length tests rather than the 50cm draw length tests as one might expect. The exception is the top helmet region where the highest mean value is found in the 50cm draw length test data. From this trend we can infer that the device works at least at a very basic level of being able to output values that correspond to different impact forces.

IV. DISCUSSION

As indicated in the previous section, the output of the impact-sensing device does not provide consistent data when comparing the output values of the primary axis for each helmet region. This could lead us to believe the device does not work properly or does not provide measurements with enough precision to accurately reflect the actual force of impacts on the helmet. While this is not an incorrect inference, if we look at the output data of the secondary axes in relation to the data from the primary axis, there are some instances that suggest a possible solution to our data measurement problem.

Consider the highlighted rows in the 50cm draw length section of Table 1. The first highlighted row shows that the X-axis value, the primary impact value, is only 16*g*. While we would expect impact values at the highest draw length to be relatively high, this value is actually the lowest primary impact value for this helmet region. However, the associated secondary impact values are much higher than expected. The Y-axis value is 63*g* and the Z-axis value is 23*g*. Similarly, the other highlighted row has a low primary impact value of 32*g* in the X-axis and higher than expected secondary impact values in the Y and Z-axes, 47*g* and 68*g* respectively.

These deviations from expected output are not isolated to the facemask region output shown in Table 1. These data relationships are important because they show that the lower than expected output values on the primary impact axis are not necessarily the result of a device malfunction. Rather, it appears that the force intended for the primary axis was sensed on the other two axes. This leads to speculation that using a single accelerometer limits directional sensitivity of our device.

We determined that the directional sensitivity of our single accelerometer is such that the small variation in the consistency of impact location provided by the ballista machine can cause our device to produce the output we see in these instances. If the ballista bolt deviates even slightly from the center of the primary impact point on the helmet, the output data reflects a lower than expected primary axis output and higher than expected secondary axes outputs. We believe this problem could be solved by placing multiple accelerometers around the perimeter of the helmet. This would improve the accuracy of our sensor device by increasing the number of direct impact measurement points distributed over the helmet regions.

V. Conclusions

After reviewing our data, we concluded that the system works reasonably well for a rapidly developed prototype. However, our hypothesis must be rejected since our output data indicates a lack of consistency in sensed impacts from one helmet region to another. The primary drawback is the limitation imposed by using only a single accelerometer for sensing impact events. We noticed that when the ballista bolt deviated even a small amount from the center of the targeted impact axis of the helmet, the recorded impact data would not reflect the same output as when the bolt hit directly on the targeted axis. However, the general trend was that as the ballista draw length increased, the output data showed an increase in sensed impact force.

From these data we can conclude that the device provides basic functionality. The XBee Wireless RF network provided a low power, user-friendly means of transmitting data with the frequency, speed, and reliability required for tracking sudden impact events. The Arduino microprocessor proved to be a flexible and reliable platform for integrating the I/O components and implementing the processing algorithm.

Acknowledgment

I express my sincere thanks to Dr. Mingrui Zhang, Dr. Sudharsan Iyengar, and Dr. Narayan Debnath for their advice and guidance. Thanks to Chris Popp and Andy Tolbert for their guidance in using the XBee and Arduino in this project. Thanks to Chris Sutton for assisting in the design of the ballista device. Thanks to Peter Boysen and Tim Marshall at Enlightened Equipment for the use of their facilities for taking the photos used in this paper and associated presentations.

References

[1] Christopher Alcaraz, et al., *Head impacts during high school football: a biomechanics assessment*. Journal of Athletic Training, 44.4:342, 2009.

[2] Mayo Clinic Staff *Diseases and Conditions: Concussion* 2014. MayoClinic.org. http://www.mayoclinic.org/diseasesconditions/concussion/basics/complications/con-20019272. Accessed: March 20 2014.

[3] Julian E. Bailes, M.D., et al., *Role of subconcussion in repetitive mild traumatic brain injury*. J Neurosurg, 119:1235-1245, 2013.

[4] Thomas M. Talavage, et al., *Functionally-Detected Cogni- tive Impairment in High School Football Players Without Clinically-Diagnosed Concussion*. Journal of Neurotrauma, 30:1-12, 2013.

[5] Margolis, Michael. 2003. *Arduino Cookbook.* Sebastopol, CA: O'reilly Media, pp. 1-3, 469-472.

[6] Digi International White Paper. *Demystifying 802.15.4 and Zigbee.* Digi International Inc. 2007-2008. http://www.digi.com/pdf/wp zigbee.pdf. Accessed: February 2 2014.

[7] *XBee Multipoint RF Modules: Embedded RF Modules for OEMs.* Digi International Inc. 2006-2011. http://www.digi.com/pdf/ds XBeemultipointmodules.pdf. Accessed: February 2 2014.

[8] *Arduino XBee Shield Guide.* 2006. Sparkfun.com. https://www.sparkfun.com/tutorials/194. Accessed: February 6 2014.

[9] *Adafruit Analog Accelerometer Breakouts.* 2013. learn.adafruit.com. http://learn.adafruit.com/adafruit-analog-accelerometer-breakouts/wiring. Accessed February 6 2014.

[10] *ANALOG DEVICES Data Sheet ADXL377.* 2012. Analog Devices, Inc. Revision 0: Initial Version. September 2012.

User-Friendliness of Atlas Mapping in Family Tree Software

Chue Vang
Winona State University
Computer Science Department
Winona, United States of America
Cvang11@winona.edu

***Abstract*— There are many different family tree applications available, but few with integrated atlas mapping. The integrated atlas mapping marks areas where families have migrated to and lived in. This paper presents testing methods and surveys users to determine if they prefer to have the atlas mapping. The applications used are the Family Tree Maker and the iFamily on a MacBook Pro. The testing was done on three groups of participants. Two groups used the Family Tree Maker and one used iFamily. One of the two groups testing the Family Tree Maker will not test the atlas mapping aspect of the application. Analysis shows that atlas mapping does not make the application user-friendlier. Although the data showed that atlas mapping made the family tree application less user-friendly, the data was not significant.**

I. INTRODUCTION

There are approximately 7.2 billion people living in the world. Many of them would like to know what their background or ethnicity is. Knowing your background is important for a person when they are going through self-development [1]. People who want to know more about their family origins often don't have any information because much of it is lost over time. Though families have various ways of passing down their family heritage through dances, story telling, and various forms of media—many changes in all cultures means it is getting harder to keep track of this information. A modern way to keep track of this information is through the use of a family tree application.

A family tree application stores information of a person's family and events that have happened to them. Typical genealogical information is composed of a person's surname, first name, gender, date and places of birth, baptism, death, and burial [2]. This information is normally stored in a data structure form of a tree with a parent node and multiple child nodes [3]. Each tree arc branches from child to parent as shown in Figure 1 [3]. The root is the latest child recorded on a tree and has no other child nodes connected to it.

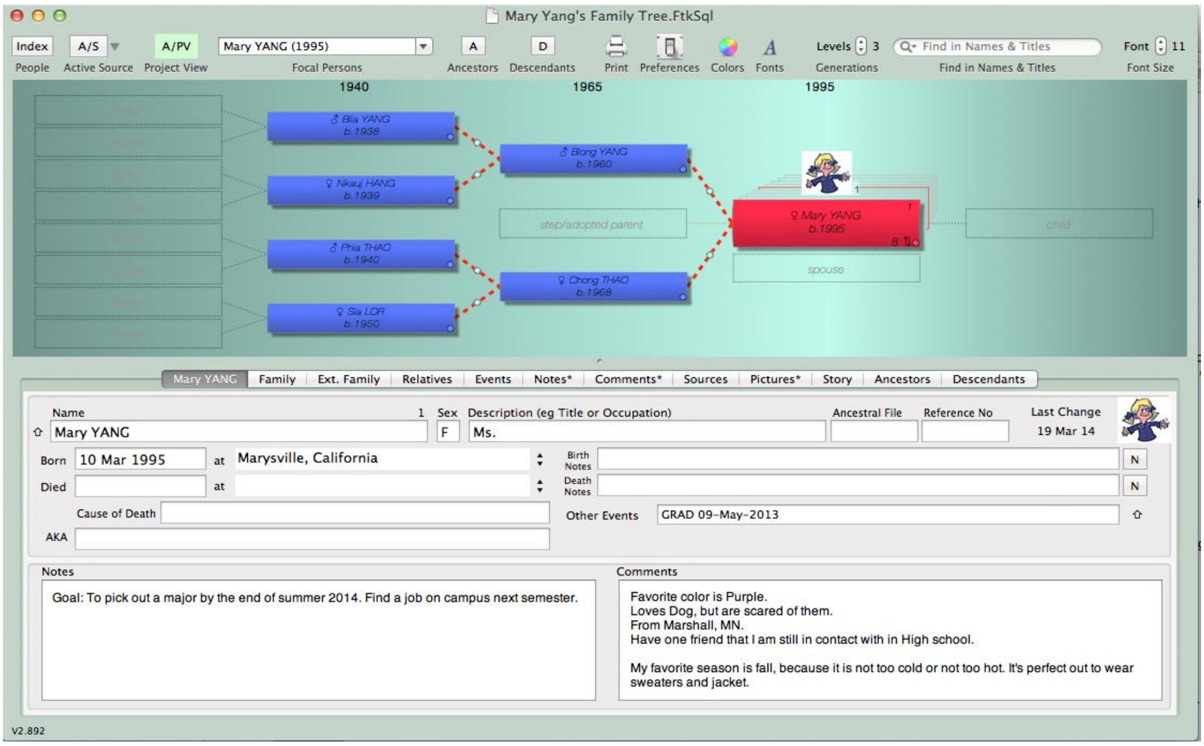


Fig. 1. Interface of iFamily Application

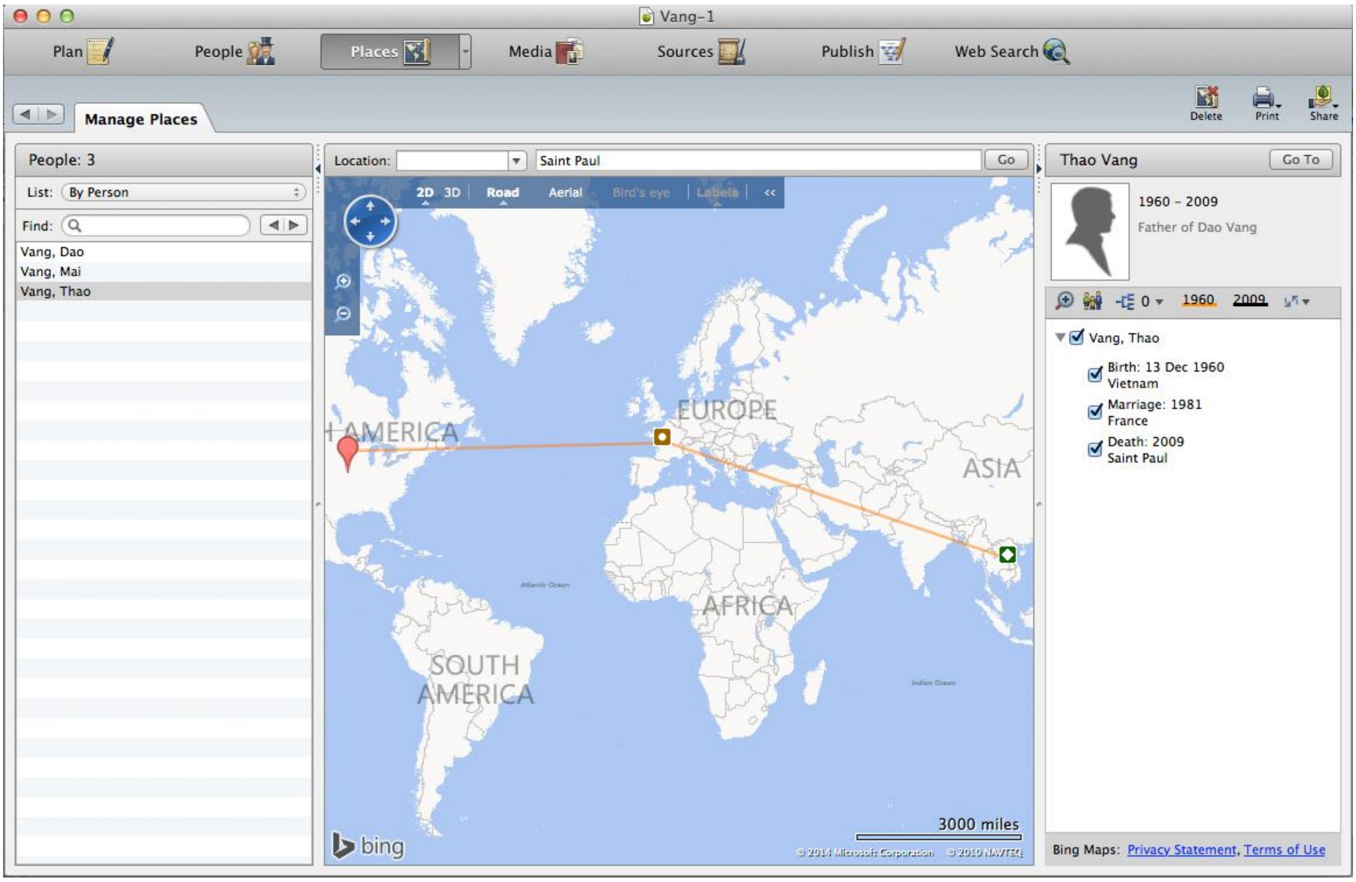


Fig. 2. Family Tree Maker - Atlas Mapping

There are many family trees that have already been designed such as Ancestry.com, Family Tree Builder, Geno Pro and more. While many of these applications are popular, there are few with atlas mapping. Atlas mapping on a family tree application is a feature that allows a person to locate areas around the globe and mark it with a pointer for where family events occurred. This function allows family members to be able to trace back their heritage and perhaps even history geographically. Atlas mapping has been used in multiple fields such as global positioning systems (gps), traffic monitoring and even for tracking Internet usage [4].

The goal of this study is to find if users would like to use the family tree application with an integrated atlas mapping to trace their family origin. The users will input the locations that their family has lived in and/or moved to, as shown in Figure 2. This study will mainly test if the users like the atlas mapping feature and to find out if it is useful to integrate in to a family tree application.

I hypothesize that a family tree application with an integrated atlas map of family history is more user-friendly than one without.

II. METHODS

The approach for the survey was a usability study test. Users completed a survey based on the application that they tested. A series of questions were then asked to the users regarding their opinion of the program.

A. *Participants*

Participants were chosen at random from a range of people on the Winona State University campus. However priority was given to participants whose ancestry originated in multiple parts of the world. This allowed the users to test the application more thoroughly and allow us to be able to gather results that are more precise. Participants were given a consent form in which they signed if they agreed to the terms and agreements. There were a total of 18 participants that conducted the usability test.

B. *Materials*

Each user was presented a MacBook Pro with the application, Family Tree Maker or iFamily. The application to be tested by the participant was chosen at random. With the MacBook Pro, the 3D feature of the atlas mapping in the application was not usable as it would be with a Windows operating system. The participants then completed a sequence of tasks, which allowed them to familiarize themselves with the application. There were three groups of participants, one group which tested the Family Tree Maker application without using the atlas mapping (Group 1), and the second group which tested the application with the use of atlas mapping (Group 2). The third group tested the iFamily application (Group 3). Each group had 6 participants to complete the usability test.

TABLE 1. Groups with applications

Group #	Application
1	Family Tree Maker (no atlas mapping)
2	Family Tree Maker (with atlas mapping)
3	iFamily

Participants in Groups 1 and 2 used the same application but Group 1 did not test the atlas mapping aspect of the application. This helps to rid of any skew/bias that may occur due to using another application that may have a different interface but no atlas mapping. As seen in Figure 3. Family Tree Maker's interface is a bit more complex than the interface of iFamily in Figure 1.

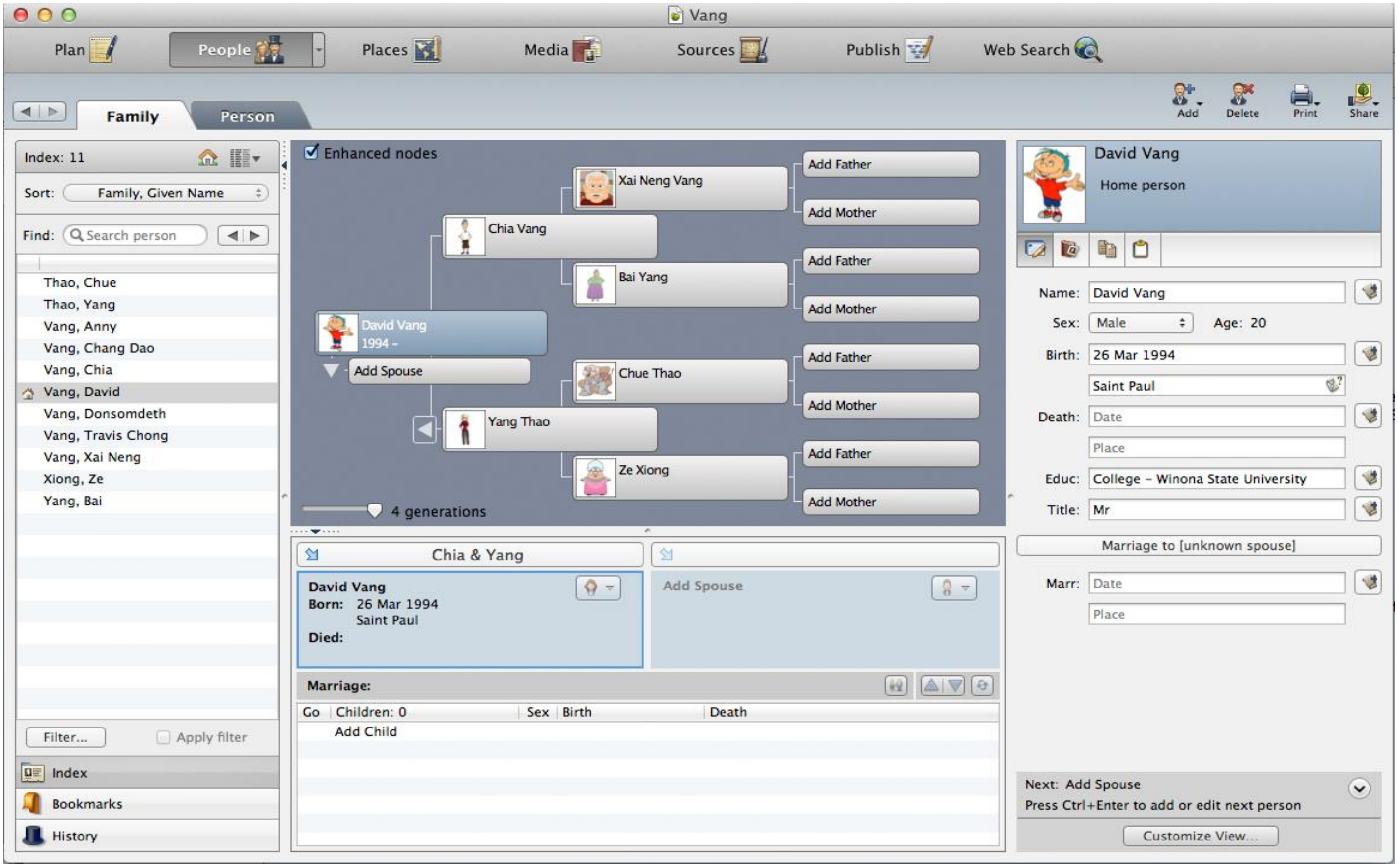


Fig. 3. Family Tree Maker

C. *Test Cases*

Each of the participants had a set of test cases to complete. Group 1 did not have to worry about the Family Tree Maker not functioning without atlas mapping because it would work normally even if the atlas mapping aspect was ignored. Group 2 participants testing atlas mapping received the same set of tasks as the Group 1 but also received another set of tasks that tested the atlas mapping aspect of application. Group 3 had a separate set of tasks to complete since they were using iFamily and not Family Tree Maker. After completion of all the test cases for the application, the participants became more familiarized with the application allowing them to take the survey.

D. Surveys

The survey questioned users' thoughts and ratings on how they felt about the family tree. The pre-test survey started with a series of pre-questions (did not have rating bar), which occurred before the test cases were given. The post-test survey after the test cases was a paper survey, which includes a rating bar of 5. 1 is the worst rating and 5 is the best rating. The survey shown in Figure 4. is a sample of the survey that the user received.

Survey

The questions are on a rating scale of 1 to 5. One being the worst and 5 being the best. If you have any additional comments, please write it.

- **How easy was the interface to navigate?**

1 2 3 4 5

Comments:

- **How good was the information storage?**

1 2 3 4 5

Comments:

- **Did the application do well displaying information stored?**

1 2 3 4 5

Comments:

- **Did you like how you can add photos to each person in the tree?**

1 2 3 4 5

Comments:

Fig. 4. Sample of Survey Questions

There were also a series of post-questions (did not have rating bar) that were administered to gain more information on what the all participants thought as. The survey was the same for all groups. Except for Group 2, who tested atlas mapping, which had one additional question which asked participants, "How did you like the Atlas Mapping?" This additional question for Group 2 is highlighted in Table 2.

TABLE 2. All Survey Questions

Pre Questions
Have you ever used a family tree application? Yes_ No_
Do you see your family using a family tree application for multiple generations?
How have you keep track of your family history or ancestry?
Post Questions
How easy was the interface to navigate?
How good was the information storage?
Did the application do well displaying information stored?
Did you like how you can add photos to each person in the tree?
How well do you like the display of the generated family chart?
How would you rate the program overall?
How did you like the Atlas Mapping?
Short Answer Post Questions
What is your overall impression to the application?
What did you like best about the application?
What would you like the see the application have/do that it doesn't already have/do?
Do you have any other final comments or questions?

The data that was collected was then compiled and the scores of the applications were compared through two groups. Group 1's data were compared to Group 2, and Group 3 was compared to Group 1. An analysis of the data gathered will help determine whether the atlas mapping is user-friendly in a family tree application. Then a comparison of Group 3's data with Group 1's data will be conducted to verify if the atlas-mapping feature is a desired feature when designing new applications, if the results show that the application is preferred. This part of the test is important for future research because to determine if the atlas mapping design is applicable for future designs.

E. T-Tests

To compare the significance of the results, a T-test was used. A T-test checks whether two sets of results are significant and large enough to say that the difference between the groups is not likely to have been a chance finding. The P-value is set to 0.05 in-order to find out if the data is significant [5]. Since atlas mapping is supposed to make the application more user-friendly, the T-test is set to a one-tailed test. The degrees of freedom is 10 with an alpha level of 0.05 giving us a cut-off of 1.812.

III. RESULTS

Figure 5. shows that Family Tree Maker with Atlas Mapping has the lowest average score of satisfaction, 3.67 out of 5. The highest was iFamily for Leopard with a 4.17 out of 5. Comparing Group 1's data to Group 2's data using a T-test showed that the difference was not significant, $t(10) = 0.12$, $p < 0.05$. Since the T-score of 0.12 is below 1.812 it shows that the data was not very significant. The same goes for the T-test of Group 2 and Group 3, $t(10) = 0.10$, $p < 0.05$. The T-score of 0.10 is below 1.812, meaning that the data isn't significant. This rating score is from the question that was asked to the participants, "How would you rate the program overall?"

In Figure 6. the data shown is very similar to Figure 5. The compiled data of Figure 6 consists of the rating from all the questions asked except its overall rating question. This data also shows results that closely resemble the previous chart. The results from this chart shows that the Family Tree Maker with atlas mapping was the lowest rating.

Again using a t-test on the results showed that the difference was not significant for Group 1 and Group 2, $t(10) = 0.02$, $p < 0.05$. Since 0.02 is below 1.812 it shows that the data was not very significant. The t-test for group 2 and group 3 was also not significant as the t-score was 0.01, $t(10) = 0.01$, $p < 0.05$.

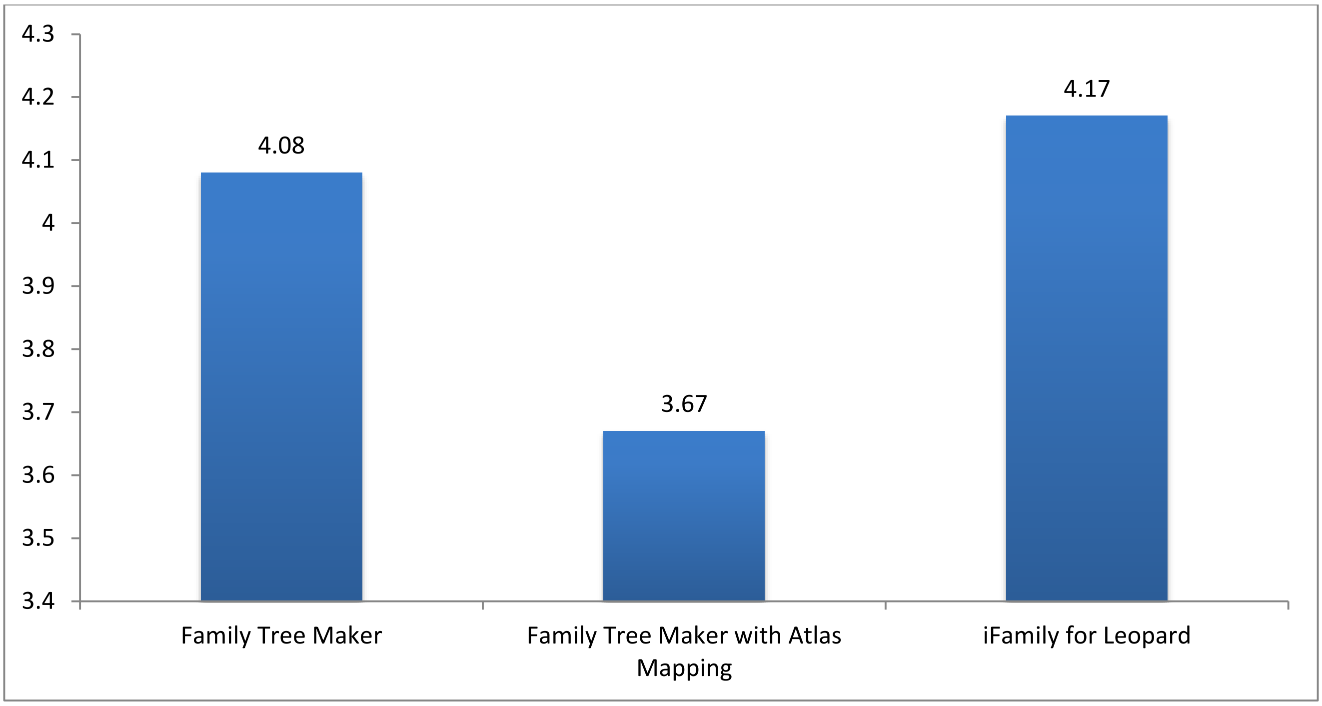


Fig. 5. Overall Average Rating out of 5

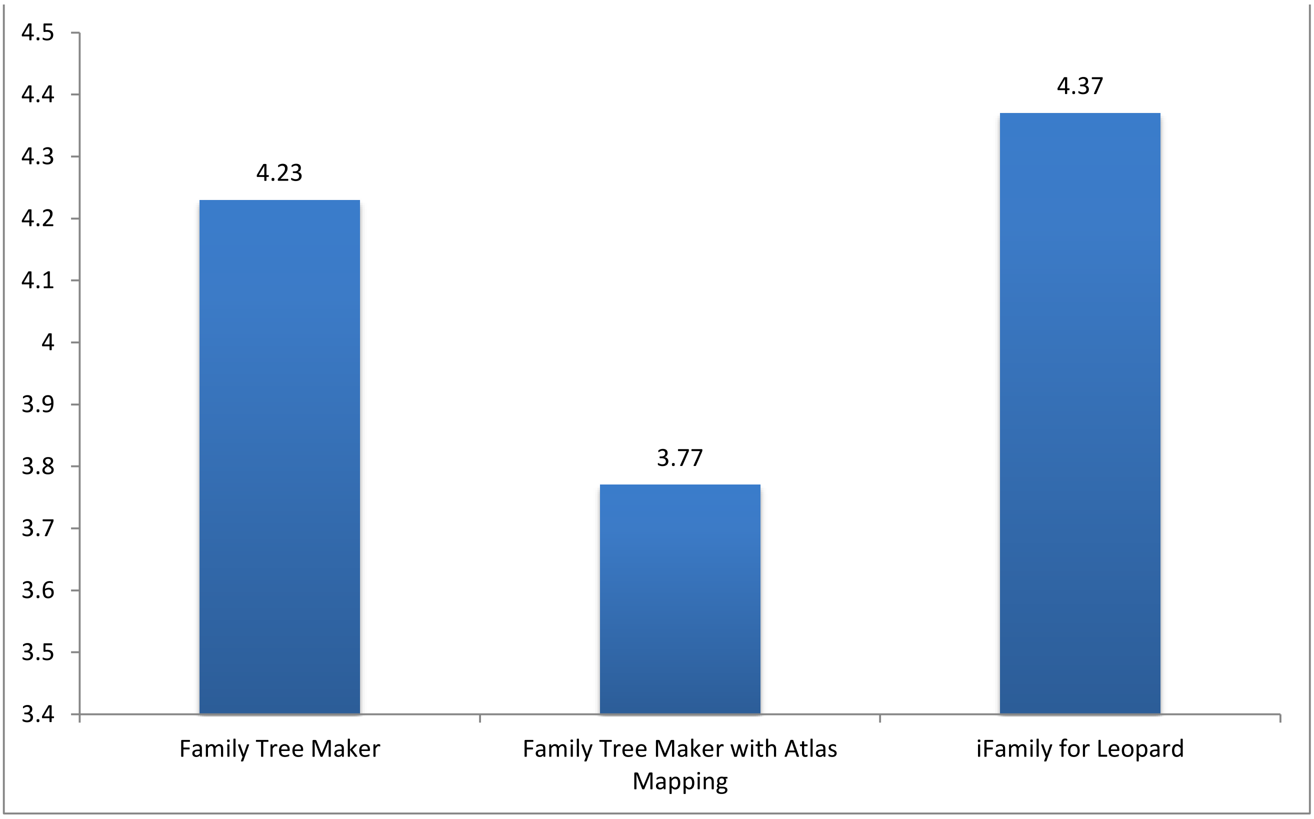


Fig. 6. All Questions Average Rating out of 5

Since the T-scores are not above the cut-off the data is not sufficient to determine if atlas mapping is negatively affecting the application's rating. If the T-scores exceeded the cut-off then it would mean that atlas mapping did have a negative effect on the application.

The t-scores shown in table 3 lists out the two groups compared, which questions, the T-score, P-value, cut-off and if the data was significant. The overall rating question is a single question that was asked and All Q's are all the questions asked excluding the overall rating question.

TABLE 3. T-tests and T-scores

Questions Compared	Groups Compared	T-score	P-value	Cut-off	Significant
Overall Rating	1 & 2	0.12	0.05	1.812	No
Overall Rating	2 & 3	0.1	0.05	1.812	No
All Qs	1 & 2	0.02	0.05	1.812	No
All Qs	2 & 3	0.01	0.05	1.812	No

The question that was asked only to group 2 was "How did you like the Atlas Mapping?" It scored a 4.17 out of 5, which resulted in an 83% satisfaction. The results from this show that users liked the feature, but from the previous results, the atlas mapping seems to bring down the user-friendliness of the application.

TABLE 4. Results of Atlas Mapping Survey Question

Participant #	1	2	3	4	5	6	Total	Average
How do you like the Atlas Mapping? (Rating 1 - Worst, 5 -best)	3	4	5	3	5	5	25	4.17

IV. CONCLUSION

Although the data collected showed that Atlas Mapping made the application less user-friendly, the data was insignificant. Thus, it cannot be concluded that atlas mapping is a negative feature to include in applications. However, the results did show that atlas mapping does have a negative impact on the user-friendliness of the application, Family Tree Maker.

V. DISCUSSION

There were several areas in which the survey could have gone better. If there were more participants to take the surveys then the data could have been more significant. There were only six participants in each group, which totaled to 18 participants. With each usability test taking about 30 – 45 minutes, it was quite time-consuming causing prospect participants to be antsy about taking the test.

The rating bar was out of 5, if I had set it to 10 it could of gave a better rating scale. With at rating bar of 5, it only gives the participants a 20% interval of judgment, where as the 10 would give it a 10% interval. With a smaller interval the data would be more precise. In one of the tests, a participant even put down a 4.5, which would be a 9 in a rating bar of 10.

One of the interesting things that I found from this study was that people were quite concerned about the photos. They liked that they could put photos and were allowed to edit them and tag it with notes. I think that this could be a potential study in the future, enhancing the photo features.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my professors Dr. Zhang, Dr Debnath, and Dr. Iyengar, for their support as I worked on my project. Without their guidance, this project would not have been possible.

In addition, I would like to acknowledge my friend Jouapag Lee for her continuous help in proofreading my material.

REFERENCES

[1] R. Parke, "Development in the family," Annual Review of Psychology, Vol 55: 365- 399, 2003.

[2] J. Wesson, M. Plessis, C. Oosthuizen, "A zoomtree interface for searching genealogical information," *AFRIGRAPH*, pp 131- 136, 2004.

[3] L. Georgiadis, *et al.* "Data structures for mergeable trees," ACM Transactions on Algorithms, Volume 7 Issue 2, pp 14:1- 14:30, 2011.

[4] R. Durairajan, "Internet atlas: a geographic database of the internet," HotPlanet '13, pp 15- 20, 2013.

[5] (2006) "The T-Test" [Online]. Available: "http://www.socialresearchmethods.net/kb/stat_t.php"