4-23-2018

Gender Differences in Motivations for Social Media Use Among College Students

William Daniel Parsons

Concerns Related to GUI design and Language Features and Tools that Address Them

By

William David Parsons

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Diploma

to
The Honors College
of
The University of Alabama in Huntsville
5/3/2017
Honors Capstone Director: Dr. Richard Coleman

Lecturer of the Computer Science Department

[signature]	5/4/2017
Student (signature)	Date
R. Coleman	5/4/2017
Director (signature)	Date
H.S. Ranganath	5/4/2017
Department Chair (signature)	Date
[signature]	5/4/17
Honors College Dean (signature)	Date

ABSTRACT

This paper is designed to be a tool for students of the CS: 499 course to be used in helping them pick languages for their projects in senior design with designing GUIs in mind. In the **GUI section** I will discuss different concerns involved in the design of GUIs as well as tools, such as Qt, that can be used to make it easier for the students to craft their programs in a given language. In the **language section** I will tackle the subject matter of each language in the order of C++, Java, JavaScript, Python, PHP, and finally Ruby. In these sections, I will discuss basic information regarding the language, as well as how each language handles databases, button creation, data storage, security risks and data encryption, and web development. Since PHP and JavaScript are languages built for web design, the web development section will not exist for them. Finally, the **conclusion** contains an overview of the concerns involved in designing GUIs as well as generalized guidelines for picking languages without knowing the specifics of the program needed to be designed.

Table of Contents

LIST OF ILLUSTRATIONS

I. INTRODUCTION

This paper was created with the purpose of educating future students of CS499 Senior Design on design needs for GUIs as well as facilitate picking a language for their project. The paper is designed to cover six different programming languages - C++, Java, JavaScript, PHP, Python, and Ruby - which have been used rather frequently in the class or are languages the students are likely to know. I will first start the discussion covering a number of the different concerns that might be involved in the design of GUIs. In this section, I will also cover tools, such as Qt, that can be used alongside languages to make it easier for the students to craft their programs. After I finish the discussion on GUI design, I will tackle the subject matter of each language in the order of C++, Java, JavaScript, Python, PHP, and Ruby. In these sections, I will first discuss basic information regarding the language. Then, I will cover how each handles databases. After that, I will discuss what it takes to create a simple widget, a button, in each of these languages so that the developer can have reference on the difficulty of performing these tasks. Next, I will cover how each of the given languages is able to store its data. That way when a project requires handling data to a high degree of specificity, the student will know how the language handles that, if it can. Then, I will cover major security risks a developer might encounter with the language in a given project, if the language has them, and how to encrypt data. Finally, I will cover what tools the language has to create web-based or internet-based application design. After I have discussed each of the individual languages, the paper will conclude with a review of what each language has to offer and which might be best for a number of example applications.

II. GUI CONCERNS and TOOLS

The GUI, or graphical user interface, of the program is the portion of the program that the user interacts with most. When a programmer or software engineer is designing a program with a GUI, there are a number of concerns involved in GUI design that they should take into consideration: data binding, structure, layout, inter-behavior, data formatting, validation, styles and themes, and security. In this section, we will discuss what each of these concerns are; later, when we discuss the languages, we will be able to determine what tools are available in each of these languages that can help us design with these concerns in mind.

2.1 DATA BINDING

Data Binding covers the basis of connecting to the data sources for whatever API constructs they have. So it refers to the “API constructs for creating and maintaining communication points in the GUI that provide bidirectional communication with the back-end logic, data model, and application services” (Mijailovic and Milicev 759). This would include the use of databases for applications that need the ability to store user information and data for multiple users. For the purposes of the language analysis, we will be focusing on databases as they tend to require

additional tools on top of the language in order to work. For example, many languages employ SQL (a database programming language) to handle database operations and interact with the server interface for their language.

2.2 STRUCTURE

Structure refers to the implementation of the GUIs structural widgets (Mijailovic and Milicev 759). These would be the buttons, scroll bars or any other interactive objects the developer wishes to place within the GUI. When crafting GUI structures programmers need to consider what kinds of structures will be best for the application, what state these structures will need to be in when the application begins, and what are the ways in which the a user can interact with these structures.

2.3 LAYOUT

Layout refers to the management of the position, size and visibility of GUI widgets (Mijailovic and Milicev 759). Programmers need to consider how the objects within their GUIs will react should the user change the window size. Should the developer be creating a web based application, he or she might also need to consider how their website will handle phones or tablet devices that try to access it. They might be able to design the website so that there are not many changes necessary, or they might need to change the layout depending between devices to make up for screen space.

2.4 INTER-BEHAVIOR

Inter-behavior handles the necessary coupling, or interactions, between GUI widgets (Mijailovic and Milicev 759). While the ideal is to minimize coupling within the application as much as possible, there will be times where the developer is unable to remove it from some of the widgets that they have created within the GUI.

2.5 DATA FORMATTING

Data formatting involves how the data is stored (Mijailovic and Milicev 760). For example, a date such as August 4, 2010 could be stored as ***4-08-2010***, ***4/08/2010***, ***8/4/2010***, ***August 4, 2010***, etc. Even within the same application, it is possible to have information stored or displayed with multiple different formats. In their article, Mijailovic and Milicev refer to data formatting as "[the] API constructs for defining format definitions for all types of data and defining applicability policies for these format definitions, i.e., for mapping between format definitions and parts of the GUI that these definitions should be applied to" (760).

2.6 VALIDATION

Validation refers to the input verification that is sometimes necessary for widgets within the GUI (Mijailovic and Milicev 760). If a programmer has created a search bar within their GUI, they

might only accept certain types of input. They might not allow the user to input characters from the Cyrillic alphabet or Chinese Kanji.

2.7 STYLE AND THEMES

Styles and themes refer the visual changes made to a GUI in order to make the GUI look and feel as desired. These refer to the colors, fonts, gradients, shadings, borders, etc. used to create a GUI (Mijailovic and Milicev 760).

2.8 SECURITY

Security is another concern involved with GUIs. It is important in some GUIs for confidential information to be input into a program. An example of this would be user login information as well as saved credit card information. For this reason, encryption is vital for programs that are handling any user information that would be considered delicate.

2.9 WEB DEVELOPMENT

As there are projects within the CS499 course that require the students to design web applications, it is important to understand the distinction between them and standalone programs. Web applications need to be able to run over the internet and be accessible by whatever users you intend to allow access to the application. This is typically done using a client server architecture, in which you create a server that holds all the page information and backend logic. Then, you access these pages on the client side (the user side) and pull them up through a series of networking requests. We will discuss in the language sections the tools available to assist each language in the creation and maintenance of web applications.

2.10 ADDITIONAL CONSIDERATIONS

There are also additional programming tools available to some languages that assist in the creation of GUIs. QT for C++ and WinBuilder for Java are examples of these tools. These tools like the one shown in figure 2-1, WinBuilder, help in the construction of GUIs by allowing a drag and drop format for creating the basic constructs within a GUI relating them as needed. We also need to consider whether or not we are creating a web application or a regular application. As senior design does not currently cover the creation of mobile apps, we will not consider mobile app design in this paper.

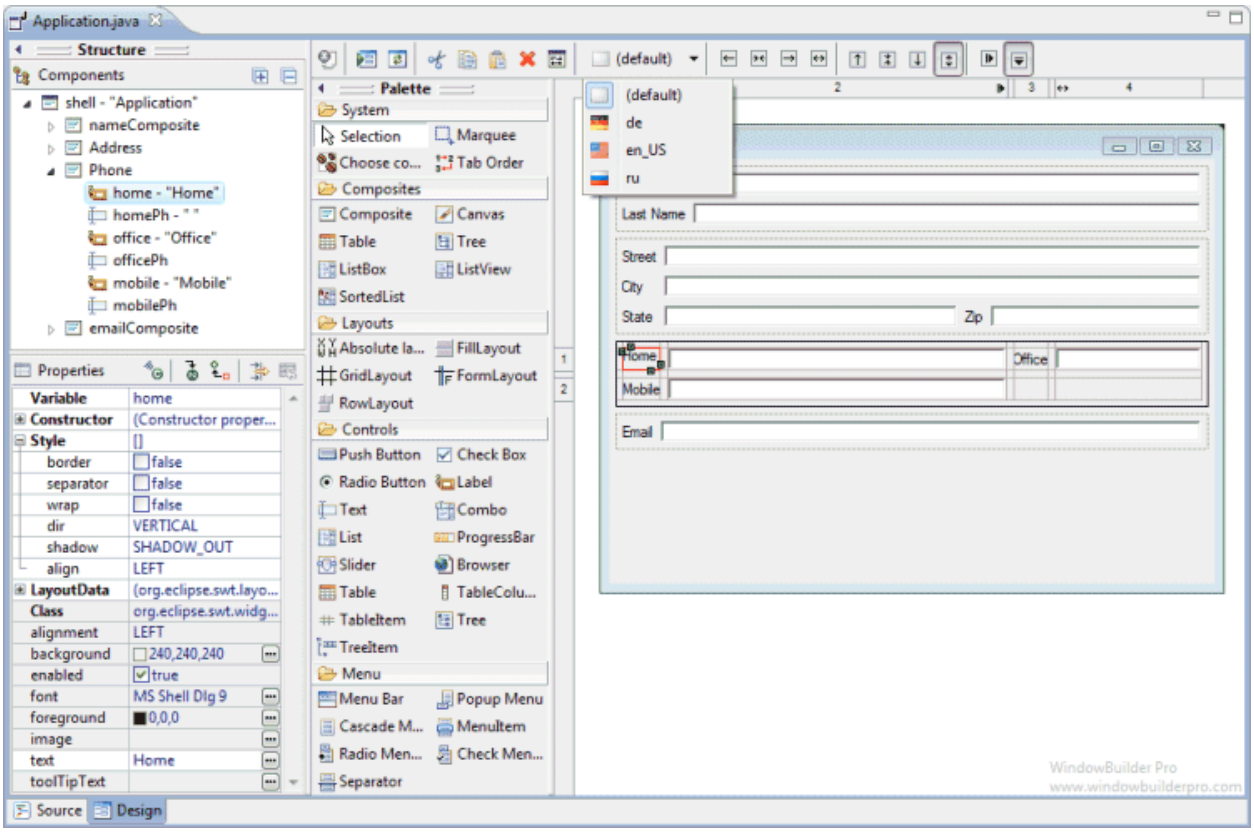

Figure 2-1. WinBuilder is a tool for java that allows the programmer to create GUIs in a drag and drop fashion, simplifying the process. Google Images.

III. C++

C++ is what is considered a compiled language. In other words, it compiles directly to a machine's native code. This means that when it is optimized, it will be able to run much faster than those that are interpreted. C++ is also a strongly-typed unsafe language. It allows for a lot of control of data types, but it also allows for a lot of programmer errors to occur. The language does not check while people are programming to see when the coders might have made a mistake.

3.1 DATABASES

C++ has the capacity to be used to create databases, but it also comes with obstacles that the developer might otherwise not have to consider if they were using an interpreted language. As Manoj Debnath, a senior technical program manager at Microsoft stated in his article, "it is the configuration issues such as importing [the] right library, drivers to use, how to access them, and so forth, that make it an uphill battle." Creating a database using C++ with a MySQL database, requires a designer to install the MySQL client library, MySQL C/C++ connector and ODBC or Open Database Connectivity drivers. So if they are looking to use C++ as their language for

coding a database, they can do it; however, I would not recommend it unless they simply wish to challenge themselves. Some of the other languages discussed later in this paper will be easier to use. Also, the speed which is gained from using C++ will not matter as much for the projects existing in the CS499 class. As it is not likely for them to be dealing with significant enough amounts of data, for example numbering in the billions of entries, the speed gained does not matter much.

It is worth noting that Qt allows the developer to bypass some of this difficulty by providing a "platform- and database- independent interface for accessing SQL databases" (Summerfield 315).

3.2. STRUCTURE AND LAYOUT

Similar to creating a database, constructing GUI widgets in C++ is difficult. Constructing C++ widgets without tools requires the designer to draw each individual object and then group them as necessary. Thankfully, C++ currently has tools readily available that assist in the creation of GUIs. Qt is one such tool. It allows a programmer to speed up the process of GUI crafting by allowing them to drag and drop basic widgets such as buttons, sliders, text boxes, etc.

3.3 DATA FORMATTING

This is a subject where C++ shines. As it is a strongly-typed language, the developers will have large amounts of control over how the data they have is stored. When they declare a type, they are dictating to the computer how much space it should have available to store data. In a strongly typed language they will have a large amount of options as to what that space is defined to be: boolean, int, short, long, float, double, char, string, etc. Programmers can specify multiple different data types for all of their information and have a lot more control than they otherwise would be able to have. For instance, if they are handling data in which accuracy and precision are vital, it would be better for them to use a language such as C++. By declaring the variables as doubles, they will have a large amount of space dedicated in the computer's memory to store their values. To contrast, in an untyped language they would declare all variables in the same manner without assigning a type determining how they would be stored. While it is possible to determine how much space this variable occupies, it will not be as simplified as it is in a strongly-typed language.

3.4 SECURITY

There are a number of encryption libraries usable for C++. One of the commonly used ones is OpenSSL. OpenSSL is an open source project that provides a number of features for Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols, as well as having a general-purpose cryptography library.

3.5 WEB DEVELOPMENT

For web development, developers have two options. First, they can utilize C++ to handle a lot of the backend data manipulation and use another language better suited for web-based applications, such as JavaScript, to handle creating the GUIs that the user will interact with. Second, they can use the web-based development library, Wt (Web Toolkit). This tool abstracts a lot of web-specific implementation details, like client-server protocols. Out of the two options, the latter seems to be the better of the two, as it will allow developers to avoid a lot of the hassles that come from needing to learn how to implement these protocols. Developers can even use HTML templates and other resources for convenience, if needed.

Overall, it would likely be better for the programmers to create their web-based application using another newer language. Similar to the reasoning behind database creation, C++ is most useful when there exists a need for speed in an application: the programmer is handling data in the billions of profiles, they need a highly optimized search engine, and they created a GUI for an embedded system. C++ would also be good if a designer wants to integrate with an existing C++ library. As the programs within the CS499 course do not require that much optimization, development time would likely be better spent using a different language.

IV. JAVA

Java is a concurrent, class-based, object-oriented language. A concurrent language is a programming language that uses language constructs for concurrency. These might be for multithreading, shared resources, futures and promises, etc. It is also a strongly and statically typed language. The language is high-level language in reference to others. This means that the details of the machine representation of code are not available through the language itself. Java also includes a garbage collector as its means of automatic storage management. This is in place of something like C++'s free and delete functions which give the programmer control of how memory management occurs during runtime. It was designed this way to avoid the type safety issues of these explicit deallocation functions. The language does not include any unsafe constructs, such as array accesses without index checking, since such unsafe constructs would cause a program to behave in an unspecified way (Gosling *et al.*).

4.1 DATABASES

Through the use of the java.sql library, it is possible to create a database in java. A program to connect to a database through java, JDBC (Java Database Connectivity program), consists of five steps.

1. Allocate a Connection object, for connecting to the database server.
2. Allocate a statement object, under the connection object created for holding a SQL command.
3. Write a SQL query and execute the query, via the Statement and Connection created.

4. Process the query result.
5. Close the statement and connection object to free up resources (Hock-Chuan).

Figure 4-1 demonstrates a code example of creating a database connection within Java.

```
import java.sql.*;   // Use 'Connection', 'Statement' and 'ResultSet' classes in java.sql package
// JDK 1.7 and above
public class JdbcSelectTest {   // Save as "JdbcSelectTest.java"
  public static void main(String[] args) {
    try (
      // Step 1: Allocate a database 'Connection' object
      Connection conn = DriverManager.getConnection(
           "jdbc:mysql://localhost:3306/ebookshop?useSSL=false", "myuser", "xxxx");
           // MySQL: "jdbc:mysql://hostname:port/databaseName", "username", "password"
      // Step 2: Allocate a 'Statement' object in the Connection
      Statement stmt = conn.createStatement();
    ) {
      // Step 3: Execute a SQL SELECT query, the query result
      //  is returned in a 'ResultSet' object.
      String strSelect = "select title, price, qty from books";
      System.out.println("The SQL query is: " + strSelect); // Echo For debugging
      System.out.println();
      ResultSet rset = stmt.executeQuery(strSelect);
      // Step 4: Process the ResultSet by scrolling the cursor forward via next().
      //  For each row, retrieve the contents of the cells with getXxx(columnName).
      System.out.println("The records selected are:");
      int rowCount = 0;
      while(rset.next()) {   // Move the cursor to the next row, return false if no more row
        String title = rset.getString("title");
        double price = rset.getDouble("price");
        int    qty   = rset.getInt("qty");
        System.out.println(title + ", " + price + ", " + qty);
        ++rowCount;
      }
      System.out.println("Total number of records = " + rowCount);
    } catch(SQLException ex) {
      ex.printStackTrace();
    }
    // Step 5: Close the resources - Done automatically by try-with-resources
  }
}
```

Figure 4-1. Code example on how to create a connection to a database with comments (Hock-Chuan).

4.2 STRUCTURE

Creating buttons and widgets within java is relatively simple. For example, creating to create a button there are four steps:

1. Declare the instance
2. Set the bounds (state where it will be placed in the window and how big it will be)
3. Add it to whatever panel within the GUI the developer wishes it to be a member of
4. Add any action listeners necessary

If they wished to speed up the process, they could also use a tool such as WinBuilder. WinBuilder gives the programmer an interface that allows them to drag and drop widgets, components, containers, and other items into a mock of their GUI. Programmers can then alter the size and placement of the widgets without having to simply type in numbers and run the program each time they want to check if their placement is correct.

4.3 DATA-FORMATTING

Like C++, Java is a statically strongly typed language. In other words, every variable and expression has a given type assigned to it that is known when the program compiles; and operations on these variables and expressions are restricted based on these types. There are two main categories of types: reference types and primitive types. **Reference types** refer to objects, arrays, classes, interfaces, etc. **Primitive types** refer to either booleans, integral types (byte, short, int, long, char), and floating point types (floats and doubles). This amount of precision allows the developers to format their data in a large variety of ways. Due to the strong typing, this will prevent them from incorrectly assigning expressions to different types and causing a loss of data. For example, let us say we created a variable x of type int. If we were to assign x to be the sum of two doubles, we would be losing the precision of the decimal places. While it is possible to get around this using static casting, it aids the programmer in not making a mistake they might otherwise have unintentionally made.

4.4 SECURITY

Java features a number of different security features. First, it has platform security. This includes a number of different built-in features enforced by the Java compiler and virtual machine. A few examples are strong data typing, automatic memory management, bytecode verification, and secure class loading. The **strong data typing** and **automatic memory management** provide an extra layer of protection from programmer errors, so that major errors such as memory corruption are much less likely to occur during a program's runtime. The **bytecode verification** makes sure that the code adheres to the JVM specification a keeps malicious code from corrupting the runtime environment. The **secure class loaders** ensure that any untrusted code is incapable of interfering with the running of other Java programs.

Second, Java has comprehensive cryptographic libraries that support a number of different sources: digital signatures, message digests, ciphers, message authentication codes, key

generators, and key factories. They also support a number of different cryptographic algorithms including RSA, DSA, AES, Triple DES, SHA, PKCS#5, RC2, and RC4.

Third, Java includes authentication and access control. It contains abstract authentication APIs that can incorporate different login mechanisms. Java also as a comprehensive API that allows developers to create applications requiring fine-grained access to security-sensitive resources. This allows for the developer to have control over the access granted to the user based on the user's identity.

Fourth, Java has secure communication APIs for communication protocols including Transport Layer Security (TLS), Secure Sockets Layer (SSL), Kerberos, and the Simple Authentication and Security Layer (SASL). This allows the program to authenticate its peers over untrusted networks, in turn protecting the privacy of any transmitted data.

Finally, Java contains public key infrastructure. In other words, Java has tools for managing keys and certificates as well as support for a number of different algorithms and features: Certificate Revocation Lists(CRLs), Certification Path Validators and Builders and On-line Certificate Status Protocol(OCSP), PCKS#11, PCKS#12, Certificate Stores (Oracle).

4.5 WEB DEVELOPMENT

There are a number of Java technologies that can be used to create web applications. I will list some of them however, this list does not dictate how many are needed to create a web application. Instead, it should be used as a point of reference. The first technology is the Java Servlet API. This allows the developer to define HTTP-specific classes. Dana Nourie describes what a servlet is in her article stating, "A servlet class extends the capabilities of servers that host applications that are accessed by way of a request-response programming model." So a programmer could have one that reads in text input and prints it to screen in a HTML page.

Another example is the JavaServer Pages Technology. This allows a designer to add snippets of servlet code into text-based documents. A JSP can be a text-based containing two types of text, either static data (which can be expressed in any text-based format such as HTML, Wireless Markup Language, or XML) or JSP technology elements (that determine how the page constructs the dynamic content). For example, a developer could use the JavaServer Pages to create a number of different pages that all refer to the same header.html file, thus all having the same headings. This permits a programmer to make changes within the one file to affect all of the headers within all of the pages instead of having to redefine it for every page. The packages used in creating a JSP page are javax.el, javax.servlet.jsp, javax.servlet.jsp.el, and javax.servlet.jsp.tagext, though the program will not necessarily need to use all of them.

JavaServer Pages also has an extension, the Standard Tag Library, which allows the programmer to use a single set of standardized tags instead of applying different tags from different vendors. Other options include JavaServer Pages Standard Tag Library, JavaServer Faces, Java Message Service API, JavaMail API and JavaBeans Activation Framework, Java API for XML Processing, JDBC API, Java Persistence API, and Java Naming and Directory Interface.

V. JAVASCRIPT

Javascript is a high-level, dynamic, untyped, interpreted language. A dynamic language, a type of high-level programming language executes many of the common programming behaviors that static languages would perform during compilation at runtime. An untyped language means that the language does not have any kind of type declaration. For example, the developer would not need to specify whether the data was to be a string or an integer. Finally, an interpreted language is one where most of the implementations execute instructions directly, without previously compiling a program into machine-language instructions. The majority of modern websites use JavaScript and a lot of web browsers, on desktops, game consoles, and smartphones, include interpreters for the language (Flanagan 1).

5.1 DATABASES

One of the ways designers can incorporate databases into their JavaScript program is to use the JSDB (JavaScript Database). It can be used as a database interface, XML processor, and an internet-oriented scripted language. It is capable of handling information in DFB/xBase, ASCII, Oracle, DB/2, Posgres, MySQL, Access, or Microsoft SQL Server Tables. JSDB can be used to translate XML into SQL statements(and vice versa), generate HTML documentation from an XML or Oracle database, download web pages, index their contents in a database, or run a web database. For reasons described within 5.3, it is generally recommended to use another language, such as PHP, to interact with databases.

5.2 STRUCTURE

Creating widgets in JavaScript is relatively simple. For example, figure 5-1 contains sample code on how to create a button with JavaScript.

```
// 1. Create the button
var button = document.createElement("button");
button.innerHTML = "Do Something";

// 2. Append somewhere
var body = document.getElementsByTagName("body")[0];
body.appendChild(button);

// 3. Add event handler
button.addEventListener ("click", function() {
 alert("did something");
});
```

Figure 5-1 - Code to create a button with JavaScript

The first step creates the button object in the GUI naming it “Do Something”. The second step adds the button element into the GUI at the area specified, body is defined elsewhere in a CSS file. Finally, an event listener is added to allow the button to perform an action when it is clicked.

5.3 DATA-FORMATTING

As JavaScript is an untyped language, developers do not necessarily have as much direct control as to how the information is stored in memory. Every number within JavaScript is represented by floating-point values. This means that it can still have as much or as little accuracy as desired by the designer. JavaScript is able to accurately represent fractions such as1/2, 1/8, and 1/1024. However, it needs to approximate the value for decimals such as 1/10. This means that it is possible for errors to occur in calculations. Figure 5-2 demonstrates an example of code where this problem is shown.

```
var x = .3 - .2  //thirty cents minus 20 cents
var y = .2 - .1  //twenty cents minus 10 cents
x == y           //=> false: the two values are not equivalent
x == .1          //=> false: .3 - .2 is not equal to .1
y == .1          // => true: .2 -.1 is equal to .1
```

Figure 5-2. Code example demonstrating how an error could occur due to approximations in decimal values. (Flanagan 35)

In figure 5-2, x could be equal to .9999 instead of .1 which is enough of a distinction to problems if you try compare these values for equality. JavaScript is not the only language with this issue. Any language that uses binary floating-point numbers would also have this issue. A workaround is to use scaled integers to perform operations like financial calculations. Instead of using floating-point values, the program would manipulate monetary values as integer cents (Flanagan 31-35).

5.4 SECURITY

By default, there are several features of JavaScript that help protect users from malicious Web sites. For example, JavaScript scripts are incapable of reading or writing files on the user’s computer. Another example is that the scripts are allowed to interact with other pages within a frameset only if they originate from the same website. While these are great, there are a number of security issues that arise from using JavaScript. As it is designed as an open scripting language, it is possible for users to be able to read the scripts that the developer have created on the developer’s website. What this means is that there are many ways for malicious users to create their own scripts that write to the programmer’s scripts and steal information from the user. It would be best to use SSL/TLS (HTTPS) and place all of the checks on the server (Wilton-Jones).

VI. PYTHON

Python is a high-level interpreted language that features a dynamic type system. Dynamic typing is the process of verifying the type safety of a program during its runtime instead of when it compiles. It does this by associating each of its runtime objects with a type tag which can be checked against. Unlike most of the other languages discussed there are two versions of python that can be used, the version 2.X series and the version 3.X series. 2.7, the current and last iteration of the 2.X series, is the older of the two and is still within wide use despite its age. There are a large number of applications using it so legacy code is a factor. Since it has reached the end of its versioning some developers consider that as a positive as that means they do not have to keep up with all of the updates made to the language anymore. The 3.X series has newer features unavailable within the 2.X series and is still being updated new versions. Either are acceptable and which a designer chooses will be dependent on their application's needs (Lutz xxxvii).

6.1 DATABASES

The Python standard for database interfaces is the Python DB-API, described by PEP 249. There are a large number of databases supported within Python: Informix, Ingres, MySQL, Oracle, PostgreSQL, Microsoft Access, Teradata, IBM Netezza, asql, SQLite, MetaKit, ZOBD, Durus, 4Suite server, neo4j, SnakeSQL, etc.

6.2 STRUCTURE

The Python library for creating widgets, such as buttons, is Tkinter. To create a button using this library is rather simple. Figure 6-1 demonstrates the schematic for creating a button.

```
Button = Tkinter.Button(master, option=value, … )
```

Figure 5-1. Tkinter button call schematic.

The parameters for the button object are master, as well as any number of options the developer wishes to specify. Master refers to the parent window to which the button will belong. Next, the programmer just adjusts any of the button options they wish to alter. For example, the developer could have command = myFunction as one of the options, where command is the name of the option and myFunction is the name of the function to be called when the button is clicked.

6.3 DATA-FORMATTING

Python is a strongly-typed language, thus forbidding operations that are not well defined, like trying to add a number to a string. Python also permits its users to create their own types via classes. Due to this a number of different type/class modules exist. An example of this is the

Decimal type/class in the module decimal which provides decimal floating point numbers to arbitrary precision and several rounding modes (Batista).

6.4 SECURITY

There are a few libraries available to be used for encryption with Python. The first is Simple Crypt. It allows for the encryption and decryption of data given a password and the data desired to be encrypted. As a note, this library also requires the pycrypto library be used. The second option is the cryptography library. This library provides high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

6.5 WEB DEVELOPMENT

Python contains numerous internet modules that allow it to perform various networking tasks: the scripts can communicate over sockets, extract form information sent to server-side CGI scripts, transfer files by FTP, parse and generate XML, and JSON documents, and more. There are also a plethora of third-party tools available to create Web-based programming in Python. For example, there is the HTMLGen system which generates HTML files from Python class-based descriptions. On top of these tools are web development framework packages, such as Django, TurboGears, web2py, Pylons, Zope, and WebWare, which support the construction of websites within Python. Some of the features of these packages include object-relational mappers, a Model/View/Controller architecture, server-side scripting and templating, and Ajax support (Lutz 12).

VII. PHP

PHP is a language primarily designed to be used in web design. Unlike JavaScript, which can be run client-side, PHP is run only on the server side. This means once the program has generated the page and sent it to the user PHP no longer runs any more actions. One of the other major features of PHP is that it is cross-platform. This means that it will run on MacOS, Windows, and Linux operating systems without much difficulty. As a note, PHP can refer to either the language or the engine. The difference is that the engine is what follows the instructions in the PHP programs that are written by the designer (Sklar 1-6)

7.1 DATABASES

There is a free software tool called phpMyAdmin for PHP which was written to aid in administering MySQL over the web. Some of the features that phpMyAdmin espouses include

- Support for most MySQL features
    - browse and drop databases, tables, views, fields and indexes
    - create, copy, drop, rename and alter databases, tables, fields and indexes
    - maintenance server, databases and tables, with proposals on server configuration

    - execute, edit and bookmark any SQL-statement, even batch-queries
    - manage MySQL user accounts and privileges
    - manage stored procedures and triggers
- Importing data from CSV and SQL
- Exporting data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocumentText and Spreadsheet, etc.
- Administering multiple servers
- Creating complex queries using Query-by-Example

It is also possible to directly incorporate your PHP with the SQL or other relational database without using the phpMyAdmin tool (phpMyAdmin).

7.2 STRUCTURE

Creating a button within PHP is rather simple. Figure 7-1 demonstrates how to create a button how to do so.

```
Resource newt_button (int $left, int $top, string $text)
```

Figure 7-1. Format for creating a button in PHP. Retrieved from the manual page of newt_button.

In this code to create a button left refers to the x-coordinate , top refers to the y coordinate and the text string refers to whatever text you wish to appear within your button. The difficulty comes in adding an event listener to this button. There are a few options for creating a button event listener. First, the programmer could inject some JavaScript code that can listen for the button click event. On the click, the JavaScript would point to the PHP script that should be run on the click. Another method would be to create a dynamic button using PHP and HTML. In PHP the designer would define the functions for when the button is clicked and the reference to the button. Then, within their HTML they would add the function calls as elements to the buttons such that when called they will create a POST, or other rest calls, to the server to call up the updated web page. Which method the software engineer wishes to use is up to them.

7.3 DATA-FORMATTING

PHP is capable of handling strings, integers and floating point values, as most languages are. Similar to JavaScript how it stores the floating point values may differ. For example, it might store 36.7 as 36.699999999999999. The maximum values that can be stored within these variables also differs depending on the operating system the PHP engine is running on.(Sklar 30) Due to its nature, PHP is also able to store HTML scripts and forms within variables.

7.4 SECURITY

PHP offers a lot of features to help keep a designer's program secure. Some of the security features and design decisions that are discussed within the PHP manual include: session security, file system security, database security, etc. Session security is needed to ensure that you control

the confidentiality of the session, as it is not possible to guarantee that the information you store in a session is only viewed by the user who created the session. There are a number of ways that your session ID information can be leaked to third parties: JavaScript injections, session ID in URLs, packet sniffing, or physical access to the device. A solution could be to implement SSL/TLS on your server and make it a requirement for users.

File system security is another of the many concerns within PHP web programming. PHP is dependent on whatever security is built into the server systems in regards to permissions on a file and directory basis. Since PHP was designed to allow the user access to the file system it is possible to write PHP scripts that will read system files such as /etc/passwd, modify your Ethernet connections, send printer jobs out, etc. So when designing a PHP program the programmer will need to make sure they have adequate checks to prevent any malicious scripts from exploiting any security measures, or lack thereof, that you might have.

Finally, there is database security. One of the best ways to secure your data within your database is encrypted. Mcrypt and Mhash are two PHP extensions that you can use to help encrypt your data before inserting it into the database and decrypt it when retrieving the data from the database. Instead of storing the password in the database you could instead simply store its hash. Then on decryption you would regain what the original password was. Another method to protect your data with databases is to establish your connections over SSL to encrypt your communications to and from the server. These are just some of the considerations that a developer would need to keep in mind when designing their web pages with PHP. For more information (on these as well as other categories), a developer can go to the PHP manual cited within the works cited page at the end of the paper. It is a good starting place for discovering the areas in which security issues or leaks might be possible within any given PHP program.

VIII. RUBY

Ruby is an open source, object-oriented programming language. It is capable of being run cross platform, has automatic garbage collection, is (mostly) portable, and supports multitasking. Ruby is an interpreted, reflective, dynamically typed language as well. Similar to python ruby has a built in interpreter that can allow you to experiment with ruby code in an environment where you can get instant feedback. (Cooper 417-420)

8.1 DATABASES

There are numerous different database systems you could use within ruby. If you wanted to create something similar to a database without the actual database you could use CSV files or YAML code. CSVs or comma separated values are text files where each line would be a “row” and the data for each “column” are, as the name would suggest, separated by commas. This will work for data sets that are relatively small. These methods contain a few problems though. They are not reliable if multiple processes are using them at the same time, and they are slow. Instead it is often recommended to use a relational database. MySQL, PostgreSQL, and SQLite are all supported within ruby. SQLite, unlike MySQL and PostgreSQL, does not run on a server. This

means that it will run on your host computer. It is still relatively fast and reliable despite this and a good choice if you just need a local database. (Cooper 204-212)

8.2 STRUCTURE

Ruby is capable of creating all the normal widgets needed for GUI development: buttons, radio buttons, checkboxes, text fields, etc. Creating a button in ruby is simple but looks a bit different syntactically from the other languages discussed. The basic syntax is

```
TkButton.new(root) {
        ……. Standard Options….
        ……. Widget-specific Options….
}
```

Figure 8-1. Basic form for a button within ruby. From https://www.tutorialspoint.com/ruby/ruby_tk_button.htm.

There are numerous standard options for a button object within ruby including activebackground, activeforeground, anchor, cursor, font, foreground, image, etc. Widget specific options include command (object invoked when mouse button released over the button window), width, state (normal, active, disabled), etc. An example of a defined button is shown in figure 8-2.

```
require 'tk'

def my proc
  puts "the user says OK."
  exit
end

root = TkRoot.new
btn_OK = TkButton.new(root) do
  text "OK"
  borderwidth 5
  underline 0
  state "normal"
  cursor "watch"
  font TkFont.new('times 20 bold')
  foreground "red"
  command (proc {myproc})
  pack("side" => "right", "padx"=> "50", "pady" => "10")
end
Tk.mainloop
```

Figure 8-2. Figure showing an example of the filled out definition of a button. Code example from https://www.tutorialspoint.com/ruby/ruby_tk_button.htm

8.3 DATA-FORMATTING

Ruby is a pure object-oriented language. The difference between this and an ordinary object-oriented language, like C++, is that everything within ruby is an object. In ruby there are no types, just classes. Practically speaking there is not that much difference between their use within ruby, except the classes that are considered types normally (strings, ints, etc.) have methods which can be invoked directly.

8.4 SECURITY

Ruby has the OpenSSL::Cipher library available for encryption and decryption algorithms though the ones available within the library are dependent on the version of OpenSSL is on the programmer's computer. If you are creating a web application ruby on rails, a program to aid in web development with ruby, has a web page at http://guides.rubyonrails.org/security.html describing the ruby-specific solutions to many of the security issues described within the PHP security section of this paper.

8.5 WEB DEVELOPMENT

As mentioned in the previous section, ruby has two applications available to it: Ruby on Rails and Sinatra. Ruby on Rails is considered to be the more popular of the two. Some examples of known sites that use it are GitHub and Twitter. Ruby on Rails is an open source web application development framework. It associates a lot of components with the sections of the model-view-controller panel. Models are used to represent forms of data used by the application and contain the logic to manipulate the data, classes would be an example. Views are the schematics for what the users see in their web browsers. It is more common to limit the amount of ruby within the view using HTML, XML, RSS, or other similar formats to define these views. Finally, controllers are the logic that connect the models and the views together. These controllers contain methods such as "show", "hide", "view", "delete", etc. to represent the actions important to that controller. Sinatra is a lightweight framework designed to help quickly create web applications. As opposed to Ruby on Rails, Sinatra does not enforce MVC or REST API and is sometimes referred to as "a library that offers HTTP deployment functionality (Cooper 305-306, 333).

IX. CONCLUSION

When deciding a programming language there will be a lot of things developers will need to consider. Are they working on a preexisting piece of software or starting from the ground up? Are they creating a web application? Will their application benefit or require a database? What kinds of security concerns will be involved with the basic architecture of the application regardless of language? What security concerns do certain languages bring with them? How fast does the application need to run? How much data will the application need to handle? Does the application need to be cross-platform?

There are some guidelines I can suggest though without knowing the answer to these questions. First, if you are creating a web application developers should avoid C++ as the difficulty curve of learning it and the tools to assist is likely greater than just learning a programming language that is better suited for web design, such as PHP or JavaScript. If your application needs to run copious amounts of data then it might be worth it to either write your application in C++ or write the code that handles the bottlenecked operations in C++. If you are creating a basic GUI where you do not need maximum optimization then either Java, Python, or Ruby would be good choices for languages, as GUI creation is relatively easy to do within the languages due to their extensive libraries. It might even be better, for those with minimal GUI building experience, to use a language like Java due to its access to tools such as WinBuilder. This allows the developer to create GUIs using a drag and drop interface and play with the GUIs without the need to rerun the program each time they wish to see the changes in positioning and graphic details adjusted.

So, what is the best language in regards to GUI design? Well, the answer is that there really isn't one. What determines the "best" language is completely dependent on the answers to the questions above and is completely dependent on a given teams experience and capabilities. For example, if you are designing a web application PHP and JavaScript would be good languages to consider as they are designed to create web applications. However, if the team has significant experience with a language like ruby or python which have tools to assist in creating web applications it might be worth considering using them instead.

Aside from these suggestions the best thing a developer can do before creating their application is to look at the data relevant to their desired application within this paper and look at the resources available to the language to determine which would be best for them.

WORKS CITED

Achour, Mehdi, et al. *PHP Manual*. *Php.* The PHP Group, 2001. Web. March 4, 2017.

Batista, Fecundio. “PEP 327 -- Decimal Data Types”. *python.* Python Software Foundation. 17 October 2003. Web. 5 April 2017

Cooper, Peter. *Beginning Ruby: From Novice to Profession*. Third Edition. New York, New York: Apress Media, LLC, 2016. Print.

Debnath, Manoj. *Database Programming with C/C++*. *Codeguru*. July 11, 2016. Web. March 15, 2017.

Flanagan, David. *JavaScript:The Definitive Guide*. Sixth Edition. Sebastopol, California: O’Reilly Media Inc., 2011. Print.

Gosling, James , *et al. The Java Language Specification: Java SE 8 Edition*. Redwood City, California: Oracle America, Inc., 2015. Web. 10 March 2017.

Hock-Chuan, Chua. “Java Tutorial: An Introduction to Java Database (JDBC) Programming by Examples”. *ntu.* Nanyang Technological University, January 2017. Web. 5 April 2017.

Lutz, Mark. *Learning Python*. Fifth Edition. Sebastopol, California: O’Reilly Media Inc., 2013. Print.

Mijailovic, Zarko and Dragan Milicev. “Empirical Analysis of GUI Programming Concerns.” *International Journal of Human-Computer Studies*. 72.10-11: 751-771. *ScienceDirect*. Web. March 1, 2017.

Nourie, Dana. “Java Technologies for Web Applications”. *Oracle*. Oracle America, Inc, November 2006. Web. 4 April 2017.

Sklar, David. *Learning PHP: A Gentle Introduction to the Web’s Most Popular Language.* First Edition. Sebastopol, California: O’Reilly Media Inc., 2016. Print.

Oracle. “Java SE Security”. *Oracle*. Oracle America, Inc. Web. 3 April 2017.

phpMyAdmin development team. “Introduction” *phpMyAdmin* phpMyAdmin, 2012. Web. 3 April 2017

Summerfield, Blanchette. *C++ GUI Programming with QT 4.* Second Edition. Westford, Massachusetts: Courier Westford, November 2014. Print.

Wilton-Jones, Mark. “Security”. *howtocreate.* 19 March 2011. Web. 5 April 2017.