Graphical Environment Tool for Development versus Non Graphical Development Tool

S.Daniel and P.Filmore

Centre for Information Security and Network Research,
University of Plymouth, Plymouth, United Kingdom
e-mail: info@cscan.org

Abstract

This paper highlights the differences, advantages and drawbacks of a graphical environment tool for development (LabVIEW) against a non graphical development tool (Java) a text base programming language. The study is centred on the developments and analysis of a sever-client application using these different technologies. To understand the differences between the different technologies, it's really useful to come back to the origins of the human computer interaction and look the differences between the different interfaces. This paper matches then some of the advantage and disadvantage for using these different technologies. It is found that JAVA has advantages in resources as it gives smaller lighter source code and LabVIEW has advantages in time, it is faster and easier to program.

Keywords

Internet, client-server, command line, graphical user, LabVIEW

1 Introduction

First of all, this paper aim to review the difference between a graphical user interface (GUI) programming tool against any other type of programming tools; this will be reviewed around a web application by building a simple client-server application.

In order to compare this technology we must come back to the source; the user interface. It's not the first time that graphical user interface has been reviewed. Since the very beginning of computer, human computer interaction (HCI) has a key role in the computer usability.

The first interface to be developed was the command line interface (CLI) which is a text based environment totally neutral. Its successor; the graphical user interface which replaces most of the command by graphical icon/button reflecting a real life environment. This was followed by natural user interface (NUI) which adds physical interaction with the graphical user interface (ex: Microsoft Surface). Qt this point of time, organic user interface (OUI) claims to adapt themselves to the current application (Chapman, S., 2008).

	Metaphor	Relationship	Control	Flexibility	Behaviour
CLI	Textual	Abstract	Directed	High	Static
GUI	Graphical	Indirect	Exploratory	High-Medium	Dynamic
NUI	Physical	Direct	Contextual	Low	Realistic

Key examples: CLI → Microsoft DOS
GUI → Microsoft Vista
NUI → Microsoft Surface

Table 1: The difference between interfaces (Daniel Makoski, 2008):

1.1 From a user point of view graphical user interface are very common nowadays:

Every computer gets now graphical user interface, even mobile devices have their own graphical user interface. These interface have been develop to help user to match their personal behaviours to the computer by association of symbols, the computer interface looks pretty much as a virtual office you can find the same kind of workspace with the tool used in real life also imitated to fit the needs of computing.

Most of the advantage and drawback have already been surrounded for the basic users.

Makoski (2008) has identified major Key points between command line interface and graphical user interface. These are:

Ease of use New user will have facilities to get into the graphical user interface as it tries to fit the behaviours at the opposite the command line interface will be much difficult to use as you need to memorize the command needed for your action that you intend to do.

Control Even with all the buttons we would, the command line interface offer more control for advance user over the graphical user interface.

Multitasking That is where the graphical user interface gets most of their power, you can display as many information as you want on your display and organize them as you want. It's also possible to control multiple objects at once.

Speed Because the graphical user interface needs to point and click with the mouse the graphical element you want to use, it seems to be slower than the command line when you only need your keyboard and perform action with a single command where you may need several clicks on a graphical user interface.

Resources That isn't a secret graphical user interface needs more resources to load the graphics and manage the interaction whereas the command

line interface needs a minimum of resources to display textual information.

Scripting A command line interface enables to execute small program to automate some of their task. This feature can be find also in the graphical user interface under the name of macro which memorizes the action perform in order to automate them.

Remote access Most of the recent graphical user interface already includes remote access without the need to perform any command line.

2 Evaluation of the technologies

It is useful to compare these technologies:

a) Text based programming language:

Often shortened to code, this programming language is made of text. It could be compared to command line, each line of code represent a specific command line which is process by the compiler to provide the final application. So the developer needs to write himself all the code, text to build an application.

As reference we use Java because it's widely use over the world for creating desktop and web application and well known to be cross platform. Java application can be use as well with Linux, Mac, or Windows and can even be embedded in a web browser; it uses a run-time engine available for these different platforms.

b) Visual programming language:

Also shortened to G language for graphical language, this programming language change totally from the text based programming language as there is no code. Everything is visual, instead of command we have box and wire to interconnect these box together and make a more advance function. It is a dataflow programming; data linked the application are visually connected in the source.

So LabVIEW is the reference for this review. As well as Java it's a cross platform language using its own run-time engine for Linux, Mac, windows and web browser.

c) Client-Server application:

The client-server application is a very basic application which uses the TCP protocol to send and receive data through Internet or a network.

In both Languages Java and LabVIEW, we will produce a server application which sends some data and a client application which receives these data.

2.1 Hypotheses and Measurements

Previous studies on human computer interaction. All the key points previously highlighted are reused there for the comparison between the development languages.

2.2 Common perceptive:

1. Graphical language is easier to use.
2. Text language gives more control.
3. Graphical language allows showing more information (multitasking).
4. Text language is faster to write.
5. Graphical language is heavier.

These hypotheses have all been discovered along general usage of user interface. We are now looking forward to see if these hypotheses are also applicable to the programming languages. Point by point we are going to demonstrate each of these hypotheses to finally conclude which of them is the best and in which circumstances. For each hypothesis, we will measures and compare the result between the languages.

2.3 Methods:

1. Ease of use:
    a. Analysis of my personal experimentation.
    b. Student survey (Mark Yoder and Bruce Black, 2006).
2. Control:
    a. Number of function available.
3. Multitasking:
    a. Review of the programming environment.
4. Speed:
    a. Time to make an application.
        i. Creating source.
        ii. Compiling.
        iii. Deluging.
5. Resources:
    a. Number of lines/blocks.
    b. Source files size.
    c. Computer memory usage.

3 Demonstrations and Experimentations

Ease of use, control and multitasking:

Evaluation of the programming environment.
Working with LabVIEW feel as simple as playing Lego, you just have to pick the block you want connect it to you data and your function is done, personally I felt really confident on LabVIEW after a few hour of trainings where I was feeling a bit confuse at the same time using Java as long as you know how to program Java is

correct, but for a non programmer it we be difficult to remember every command and write the perfect syntax needed for Java. At the opposite, when a newbie in coding can get lost to remember any command in Java, that give great control over the people who masteries these language they can do much more thing than the traditional uses, in LabVIEW we hurt ourselves again the graphical wall which allow us to use only blocks already created. This last point tends to disappear as we can create our proper object, class … but it's still taking more time. Another advantage of graphical programming is the ability to see on the screen all the data that you are working on and their relation, you can easily map your idea on LabVIEW where you need to produce an algorithm even before thinking to start any coding in Java.

LabVIEW source (Figure 1) shows clearly whereas Java source (Figure 2) is less clear, we need to read the comment to understand what it is actually doing.

In another study "a study of graphical vs. textual programming for teaching DSP" Yoder M. and Black B. (2006) intend to find which of LabVIEW or MATLAB another text based programming much closer to LabVIEW in his functionality; student rather prefer to use. They made junior-level student teaching discrete-time signal processing (DSP) on both languages LabVIEW and MATLAB. "Of the 64 students that took the survey, only 3 had learned LabVIEW prior to learning MATLAB." (Yoder M. and Black B., 2006). This can be explained by the fact that MATLAB is required in some other courses.

Table 2 shows the result of this study; almost 3 to 1 student preferred to use LabVIEW. "They say it is easier to learn and more understandable." In another advanced user state "When you know what you are doing, it's much faster to type a program than to select icons from menus and point and click to connect them". (Yoder M. and Black B., 2006). This last result can be applied to most of the text based language and also to Java.

	Matlab	Either/ Neither	LabVIEW
Which language did you learn first?	60		3
Average number of quarters of experience	2.5		1.1
Which language was easier to learn?	11	12	40
Suppose you had some simple task to do, which language would be quicker to do it in?	9	8	47
Which language is better for solving signal processing problems?	14	6	44
Which language do you prefer to use?	7 – strongly 9 – somewhat	7	13 – somewhat 28 – strongly

Table 2: Some results of the student survey (Mark Yoder and Bruce Black, 2006)

Speed:

In term of time of coding this is very **variable form beginner to advance** users. So measuring the time to make the source code on both languages should not provide significant result apart of the user experience. What we can say for sure is that is still

faster to write that pointing and clicking as long as you use only your keyboard. Then when it comes interesting is for the compiling time.

In LabVIEW there is no such thing as compiling, it run the application straight from the block diagram so **no compiling time where** in Java you have to compile you code to produce the final application, this is very short few second depending of you computer but a real drawback compare to LabVIEW.

For the deluging, in both Java and LabVIEW you are able to use breakpoint to stop the application at a specific line or place and variable watcher or probes to observer the current value of data. But due to the graphical interface of LabVIEW it's much easier to **identify problem on a flowchart** than it is in a list of command. The physical positions of the different block help him to target where the problem is. So again LabVIEW seems to be much faster.

Resources:

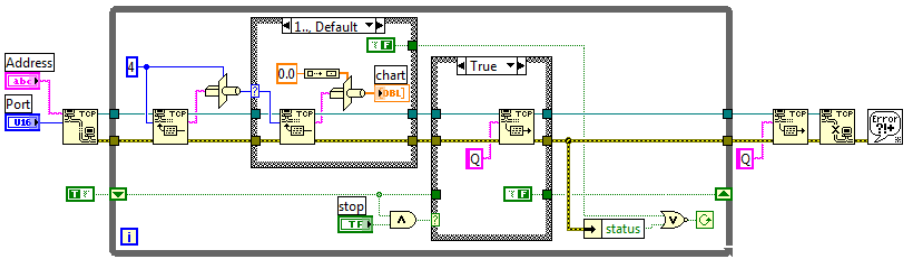

Figure 1: LabVIEW Client

```java
/**
 * Example Client program using TCP.
 */
public class Tclient {

    final static  String    serverIPname = "Localhost";       // server IP name
    final static  int       serverPort   = 3456;              // server port number

public static void main(String args[]) {

    java.net.Socket         sock = null;                      // Socket object for communicating
    java.io.PrintWriter     pw   = null;                      // socket output to server
    java.io.BufferedReader  br   = null;                      // socket input from server

    try {
          sock = new java.net.Socket(serverIPname,serverPort); // create socket and connect
          pw   = new java.io.PrintWriter(sock.getOutputStream(), true); // create reader and writer
          br   = new java.io.BufferedReader(new java.io.InputStreamReader(sock.getInputStream()));
          System.out.println("Connected to Server");
          pw.println("Message from the client");              // send msg to the server
          System.out.println("Sent message to server");
          String answer = br.readLine();                      // get data from the server
          System.out.println("Response from the server >" + answer);
          pw.close();                                         // close everything
          br.close();
          sock.close();

    } catch (Throwable e) {
          System.out.println("Error " + e.getMessage());
          e.printStackTrace();
    }
}
}
```

Figure 2: Java client

I have been able to observe on different source codes (extract: figure 1 & 2), it's quite obvious, the Java version of the client-server application is the smallest just 33 and 37 lines of Java code for this basic version without any graphical interface where LabVIEW accuse 27 and >50 blocks it's also include a small graphical interface. In term of visual space **LabVIEW seem to be again bigger than Java** to shows all the block diagrams code.

The difference become much sensitive when looking at the source file size; around 4KB for the Java source and 45KB for the LabVIEW source it's more than **ten times the size of the text based version**.

For the memory usage; Java need only the java run-time engine to run and don't need the full development kit. To work straight from the block diagram LabVIEW language need to keep the development kit running which take much more memories than a simple run-time. LabVIEW also have the possibility to build an executable application which doesn't need the development kit to work but just a **LabVIEW run-time engine similar at Java**.

3.1 Results and Comments

- ✓ LabVIEW is easier to use than JAVA or MATLAB.
  - ➢ Symbols are easier to recon than reading text.
- ✓ LabVIEW is faster to program than traditional text based language.
  - ➢ Much less error during coding (no syntax).
- ✓ JAVA takes less resources.
  - ➢ Pure text is still smaller than LabVIEW.

4 Evaluation and Conclusion

In the first part of this review we remember have seen the different key point between command line interface and graphical user interface; text based languages were supposed to have more control over the programming be faster to code and be small. And the graphical based languages were supposed to be easier to use and multitask. Finally we break the myth of text based programming is faster. That is the only real change between general interfaces a programming interface.

In the author opinion, the other drawback of LabVIEW against traditional programming is that it takes more resource and gives less control. These are not going to be some serious drawback as for the resources nowadays computers are powerful enough to run any graphical programming environment and running them without any problem. Memory isn't a problem as in the past as memory is now really cheap. The only braking point is the lack of control. LabVIEW is seriously focus on this point and are trying to give the maximum to the user and in each new version they provide more and more feature also the ability to build almost anything as our own block, library, class and much more.

Because it is particularly easy and fast to program under LabVIEW, it's really interesting to use it for **prototyping** software or any application you can just sit and start programming what in your mind and try it straight away without having to spend hours and hour to determine the perfect algorithm or debugging your application in order to make it running.

LabVIEW is a great tool for prototyping, it allows to program fast and test the application as soon as possible then we can use another language as Java or C++ to program the final application and optimize it at maximum which isn't rally the case on LabVIEW. To conclude LabVIEW is perfect to make a prototype but doesn't replace text based language as it need its own run-time less common than other languages as Java or C. LabVIEW is **complementary** to the text based language and help to **save some precious time**.

5 References

Blake J. (2009) 'Deconstructing the NUI: The Natural User Interface Revolution', *Deconstructing the NU*, blogger, [online] Available from: http://nui.joshland.org/2009/01/natural-user-interface-revolution.html (Accessed 31 March 2009).

Chapman S. (2008) 'UX Evangelist: Windows 7 NUI: Stepping Beyond the GUI', *UX Evangelist*, blogger, [online] Available from: http://uxevangelist.blogspot.com/2008/06/windows-7-nui-stepping-beyond-gui.html (Accessed 31 March 2009).

Makoski D. (2008) 'Beneath the Surface', pptx, [online] Available from: http://www.microsoft.com/surface/ (Accessed 31 March 2009).

Reyes A. (2008) *Predicting the past*, MP3, Sydney Convention Centre, [online] Available from: http://www.webdirections.org/resources/august-de-los-reyes-predicting-the-past/#description (Accessed 31 March 2009).

Sun Microsystems (n.d.) 'Writing the Server Side of a Socket', [online] Available from: http://java.sun.com/docs/books/tutorial/networking/sockets/clientServer.html (Accessed 30 March 2009).

Yoder M. and Black B. (2006) 'A Study of Graphical vs. Textual Programming for Teaching DSP', Rose-Hulman Institute of Technology, [online] Available from: http://zone.ni.com/devzone/cda/tut/p/id/3798 (Accessed 26 March 2009).

Wong W. (2006) 'Graphical-And Text-Based Programming: Complementary, Not Competitive', [online] Available from: http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=13241 (Accessed 26 March 2009).