NOVATEUR PUBLICATIONS
International Journal of Research Publications in Engineering and Technology [IJRPET]
ISSN: 2454-7875
VOLUME 3, ISSUE 4, Apr.-2017

VOICE OVER PROGRAMMING

MR. SHARDUL P. GAVANDE
Department: Master of Computer Applications Bharati Vidyapeeth's Institute of Management and Information Technology
Navi Mumbai, India shardulgavande@gmail.com

PROF. ZAHIR MULANI
Department: Master of Computer Applications Bharati Vidyapeeth's Institute of Management and Information Technology
Navi Mumbai, India zmulani8@gmail.com

ABSTRACT:

Programmers who are passionate about their work suffer from back pain and carpel tunnel syndrome. Why? It is because they are addicted to their computers and keep coding or typing for long hours. Also for physically handicapped and blind people it is difficult for them to code. So is there any solution for it? Well, there isn't any but being a programmer we can build such a tool which is completely voice based. This technology is all about simply talking to your computer to create software. Instead of typing 'n' number of line of codes we can just simply fire a command through our voice to the computer and simply develop, compile and run our software.

KEYWORDS: VOP, VSCD, SAPI, IBM.

I. INTRODUCTION:

The VOP (Voice Over Programming) System introduces an innovative method for collecting and organizing programmer codes. The VOP data entry system allows programmers to capture their verbalizations using voice manager and facilitate coding at the same time as the commands are been delivered. The verbalizations produced by the programmers appear as text in a window next to the voice space. The VOP works on the premise that as programmers can verbalize their code logic into a text format . This technique is similar to the think-aloud protocol commonly used for usability studies (Ericson & Simon, 1984). In usability studies, participants are asked to talk aloud their thoughts and tell what they are doing, why they are doing it, and what they think the result will be. This is considered a standard usability technique that is used to find out about a design simply by listening to the participant's thought process without interfering with the user's ability to carry out the task (Schneiderman, 1998).

In this paper, a result of the pilot study is carried out to evaluate the usage of the VOP, a discussion and overview of the future work. An important research activity is thus to investigate and develop such a framework which can listen to human voice command, understand it and then do the routine coding and if there occur any possible errors the system will ask verbally for its clarification. Various program understanding techniques divided into two main categories, prevenient and posterior, have been designed to address these issues and have had varying degrees of success The prevenient approach is to document the source code with programmer's understanding information along with the development of the program. The programmer understands information typically takes the form of comments, specifications and design documents, and source code descriptions.

II. LITERATURE SURVEY:

New programming development environments such as "Elucidative Programming", "Literate Programming" and Verbal Source Code Descriptor are being developed in an attempt to overcome some of these deterrents. Capturing essential understanding information from programmers and relating that information to relevant program units is the major goal of these techniques. Literate programming, introduced by Knuth considers a program and its documentation as literature, which are read by programmers in the same way as technical papers. Program fragments annotate the comments instead of the other way around (more common commenting technique). One of the major problems with this approach is that source code becomes fragmented pieces between documentation. Assembling the code for compilation, and finding and debugging syntax and run-time errors become more complicated and cumbersome processes. Elucidative programming is a variant of literate programming designed to address the problem of assembling the program fragments for compilation in literate programming. Elucidative programming attempts to solve this problem by separating the code from the documentation in a side-by-side development environment.

The Verbal Source Code Descriptor (VSCD) is a tool designed to allow programmers to document their software verbally rather than by typing them. The underlying concept is that as programmers develop code, they can verbalize their thoughts without interruption to their coding tasks. One of the unique characteristics of VSCD compared to other prevenient approaches is that the documentation and coding tasks can be done simultaneously. It is not necessary to stop programming to create the documentation. Programmers type their code and explain their coding verbally at the same time.

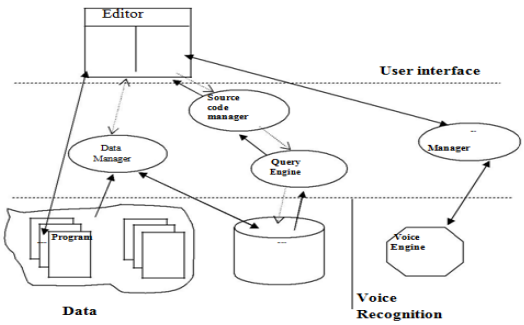
This paper contains a detailed research in depth on how a programmer can code programs by his voice commands and evaluate the effectiveness of this technique.

III. METHODOLOGY:

The design of the system is focused on enhancing the voice capturing, collecting and processing it that is commonly referred to as Voice Over Programming(VOP) which completes the desired code and it's functionality. The data entry system uses voice 3 recognition to capture programmer verbalizations while these programmers are delivering commands; an exercise that does not seem to interfere with the cognitive effort required to generate code. In the first prototype of the VOP, the IBM (International Business Machines) Via Voice engine and Microsoft SAPI (Speech Application Programming Interface) are used for voice recognition. The voice system can be manipulated (on/off/sleep) and the resulting code file can be edited from the VOP interface too. Programmers before starting need to select the mode of developing the code that is through traditional typing or by VOP. Once the voice system is in the "listen" state, all verbalization are captured, excluding the non-technical words, converting them to text and finally displaying them in the coding window of the system. The system may also provide access to the some of the standard voice recognition commands such as "Select this", "Cut this", "Copy this", "Paste this".

Two types of connecting tools are provided: connect from command to source code and connect from source code to command(for error clarification). The connection from command to source code tool captures the delivered commands, processes them by matching their associative value stored in the system and prints the code. This type of link is used when a piece of code needs to be developed from scratch. The connection from source code to command tool is the reverse order of connection from command to source code. In this, the system talks back to the user of in case of some error or syntax clarification needs to be done. This type of link could be used when a program variable or function is not properly defined and also if we want to save or run the program.

IV. ARCHITECTURAL DESIGN:



The VOP System consists of four main layers:

A. USER INTERFACE LAYER:

This provides users with tools and functions to develop source code by connecting to the voice recognition system.

B. FUNCTIONALITY LAYER:

This layer provides the core functionality of the VOP. It consists of four sub-layers.

1) DATA MANAGER: This extracts updated information of the key-value pair and stores them in the database.

2) SOURCE CODE MANAGER: When user fires voice commands the source code manager works with a query engine to respond to the user's request.

3) QUERY ENGINE: It extracts link information from the database.

4) VOICE MANAGER: It communicates with the voice engine to establish connections, provide users with voice recognition results and also terminates the connection upon request.

C. DATA LAYER:

This layer consists of all the information used by the VOP. It consists of various functions, variables, classes and related documentation. It also contains a database, which is used to store projects, files and link information.

D. VOICE RECOGNITION LAYER:

This layer includes the voice recognition engine that recognizes verbalizations made by the programmers and converts them to text.

V. ALGORITHM:

The VOP (Voice Over Programming) is a tool designed to allow programmers to develop code for their software verbally rather than by typing them. Applications involving automatic speech recognition fall into this approach which is characterized by understanding the spoken words or commands, interpreting them and performing the desired task.

A. STEP 1:

Open the VOP tool and select the language and mode of coding that is through voice commands or by traditional way.

B. STEP 2:

Deliver standard voice commands as defined by the system.

C. STEP 3:

VOP will automatically capture those voice commands, discard the non-technical words, converting them to text and finally displaying them in the coding window of the system.

- If the voice command from input frame reaches the estimated threshold value, a VOP decision (VOP = 1) is computed which declares that speech is present.
- Otherwise, a VOP decision (VOP = 0) is computed which declares the absence of speech in the input frame.

D. STEP 4:

If there occur any possibility of errors, the system will ask for further clarifications and return voice recognition results.

VI. DISCUSSION

In this research activity discussion is based on the influence of VOP on a user's experience of programming. Accordingly, the following research questions were posed as guides for our research study:

- How easy and efficient is the use of Voice Over Programming in compare to existing software development environments to develop the source code and whether it can decrease the programmer's overhead in traditional typing?
- What are the differences between the program understanding information obtained from the traditional methods of writing source code than those of VOP?

VII. FINDINGS:

As in the existing software development, programmers use to type code on their keyboard, debug and run them which would take a lot of time in thinking and writing those line of codes. Using the concept of VOP a programmer can simply develop his coding simultaneously as he processes it through his brains. It can decrease the programmer's overhead and health problems as sitting at one place continously and coding may affect but by using VOP he can avail any place nearby system an fire command to the system as we do for robots.

The difference between the traditional manner of coding and VOP is that it is completely based on voice commands. A programmer will have to speak only those commands which are defined into the system for code development. Although other non-technical words will be filtered out by the system, but it is always suggested to follow the standard way of delivering commands. An example of voice commands would be :

- **Create a method:** It will create by default method with return type true.
- **Create a class Hello:** It will create a class with name Hello along with open and closed curly braces.
- **Generate for loop for i:** It will create a for loop for variable i by default starting with 0.
- **Delete line number 20:** It will delete that particular line of code at number 20.

CONCLUSION:

This paper introduces efficient way of programming by using VOP. While a full derivation of the commands of source code for every programming language is not yet presented in this paper. An understanding of the reasons how and why a programmer faces a problem to code and how it can be resolved. VOP as a framework can also lead beneficial for physically handicapped and blind people by providing them a programming platform to develop code in an easier way. There is no report on any substantial experience with VOP in this paper. This will be done in papers after practically implementing it. With this paper only the idea and concept is introduced. Longitudinal and more extensive studies are still required to determine the usefulness of the VOP generated source code for programmers.

REFERENCES

1) IBM Embedded *ViaVoice - Providing natural, voice-based user interfaces to information and services that reside in vehicles, mobile devices and appliances*
2) Kurt Nørmark, *Requirements for an Elucidative Programming Environment.*
3) http://www.ryerson.ca/content/dam/imdc
4) https://en.wikipedia.org/wiki/Microsoft_Speech_API
5) https://en.wikipedia.org/wiki/Think_aloud_protocol.