STORAGE, RETRIEVAL AND ANALYSIS OF ACTIVE WINDOW INFORMATION

[1]Ankit Yadav , [2]Nisha Khiyani, [3]Rushabh Kabra, [4]Kartikeya Bhatnagar
Department of Computer Science,
Pune Institute of Computer Technology,
Pune, Maharashtra, India.

***Abstract*—Due to advancements in technology, many software applications are using techniques like Machine Learning, Natural Language Processing and Data Mining as a means to improve user experience. Software applications are moving towards context awareness. In this paper we focus on retrieving, storing and analyzing active window information in order to develop context aware desktop applications.**

***Index Terms*—X Windowing System, Window Manager**

I. INTRODUCTION

Linux community has a vast number of desktop environments with a motive to improve user experience. This is achieved by graphical user interface. These desktop environments enable a user to interact with applications with the help of windows. Windowing system contains window information in raw form which can be extracted and analyzed to improve user experience further. Window Manager is responsible for managing the windowing system and it also has all the meta data about window/tabs. So the active window information can be gathered by interacting with the Window Manager.[1]

Display Server interacts with the Window Manager by exchanging information in the form of atoms. So, active window information is retrieved by accessing internal active window atoms from the Display Server. This interaction between display server and user is achieved using Xlib routines.[1]

II. ARCHITECTURE

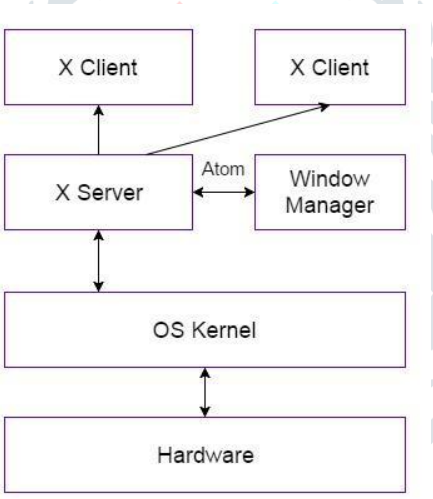

Fig -1: Architecture of Desktop Environment

There are two main components of Desktop environment of Linux:

1. Display Server
2. Window Manager

Window managers are X clients that control the appearance and behavior of the frames ("windows") where the various graphical applications are drawn. They determine the border, title bar, size, and ability to resize windows, and often provide other functionality. A display server or window server is a program whose primary task is to coordinate the input and output of its clients to and from the rest of the operating system, the hardware, and each other. The display server communicates with its clients over the display server protocol, a communications protocol, which can be network transparent or simply network-capable. Window manager and display server interact with each other in the form of predefined strings called atoms. Atoms are predefined strings which represent a property of a window or widget in the Linux desktop environment. e.g. _NET_WM_NAME is an atom which represent the property of a window title.

III. APPLICATIONS

1. Notelet: Explanation of our application.
2. Analysis of time spent on an application: This is useful to show efficiency of employees in a company. Consider an example, A company uses ‘sublime-text’ application to work on projects. By using this project one can determine how much amount of user has spent on sublime-text and for how much time he was not coding with sublime-text.
3. Application speedup: By using this project one can determine which are the applications user is using the most and then system can optimize the startup time of these applications.

IV. EXPERIMENTAL SETUP

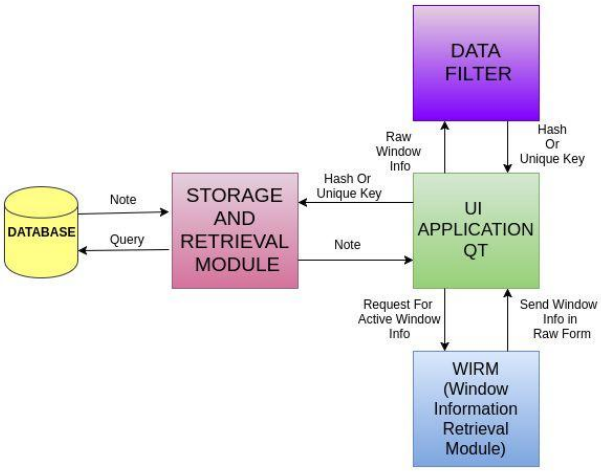

Fig -2: Architecture of Notelet

1. In this experimental setup, we describe the implementation of a Linux notes application where the user can write a note for every active window. The application is context aware and the context is the active window. This application retrieves the active window information and creates a context for the user to write the note corresponding to it.
2. This application is implemented on python. Pyxlib is used in order to interact with the display server of the system. Pyxlib is a fully functional X client library for Python programs. We are using MongoDb Database in order to store the active window information. There are four modules in this application :

A. WIRM:

- **a.** WIRM is short for Window Information Retrieval Module. Its purpose is to retrieve active window information i.e. window title and process name, etc.
- **b.** In this application, all window information is retrieved using atoms. Pyxlib library contains functions which help in interacting with the display server. Pyxlib has a structure called Xlib.display.Display which contains information about the display.[2]
- **c.** In order to retrieve active window information, active window id has to be identified. This is done by retrieving the atom identifier corresponding to the Active window, that is _NET_ACTIVE_WINDOW. Using this identifier, the active window id is retrieved. This id is used to create a window object, which is further used to retrieve the properties of this active window.We retrieve the window name and process name of the active window using atoms _NET_WM_NAME and _NET_WM_PID.[2]

B. Data Filter Module: This module takes the window information i.e. window title and process name and creates a unique hash from them so that our application can remember the context using this hash.

C. Storage Module: This module is responsible for storing the note corresponding to the hash of active window in a MongoDb Database and send it to the application whenever requested.

D. Main Application: This module handles the User Interface. We are using PyQt as the graphical user interface. This module interacts with all the remaining three modules.

V. CONCLUSION

Furtherance in Machine Learning and Data Mining are enablers for development of proactive applications. In this paper, we have described a procedure to retrieve, store and analyze the 'active window' information with a time complexity of O(log(n)) and space complexity O(n). This information can be used to develop proactive and context aware software applications.

REFERENCES

[1] B. P. Danner, A. B. Marmor-Squires, "An Advanced Process Models Application to Trusted X Window System Development", Computer Security Applications Conference, IEEE, 1990.

[2] Scott McGregor, "Designing user interface tools for the X window system",Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage , IEEE, 1989.