Next Generation Word Processors
A generalization of architecture of EasyTuteLO Extension

Tharindu Lakmal Muthugama
Department of Computer Science and Engineering,University of Moratuwa
Sri Lanka
lakmal.10@cse.mrt.ac.lk

***Abstract* — Usually the word processors are used for various documentation related activities. The subject materials in tutoring institutions are usually prepared using word processors. Currently one way of making tutorials or the documents is searching through the Internet and adding the details to the word processor separately and it has become a mostly used procedure. Adding the knowledge component to a word processor can be done in two ways where the first is to build all the knowledge within the word processor and the second is to provide a way to interact with computational and knowledge engines existing. The second approach is used in this context. The evolution of word processors can be described by several generations. If the model described in this research paper could be generalized and make a default feature, it could become the next generation word processors.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

Composing a document involves inclusion of human input and computer input together to make a fruitful output which can be read or seen. But creating documents by manipulating existing data is most seen in today's world.

This research paper is based on a project that developed the extension EasyTuteLO to the LibreOffice Writer. The extension is capable of providing mathematical formulas, solutions, alternative representations, graphs and stepwise solutions to the users who build mathematical tutorials and handouts. This helps the user to reduce the time significantly to create a better and focused subject material within few minutes. The EasyTuteLO extension is helpful for those involved in mathematics tutoring. Similarly, if there exists an extension for other fields, such as Chemistry for chemical equations, Music for musical graphs, etc., they'll use that extension. In that way the domain specific extensions can be created. There may exist some other general needs that everyone is looking for, such as translation, search, related images, related knowledge on a particular subject and language support. Some of these features can be added as default features and some of them can be kept as available features but not configured. Based on the users' requirements the additional features can be configured. This can be the gateway of a paradigm shift in word processors.

The extension architecture described here will be the base of the conceptual model

II. LITERATURE REVIEW

A. *Technologies Involved*

The following technologies were included in developing the EasyTuteLO Extension.

Python Language- Python[1] is a high-level general purpose programming language that supports functional, procedural and object oriented programming styles. Code readability and ability to implement using fewer lines of code are the advantages of Python.

UNO (Universal Network Objects) – UNO[2] is the Application programming interface used in OpenOffice.org and LibreOffice which allows to program the office software in different programming languages. UNO API supports language such as C++, Java, Python, CLI, StarBasic, JavaScript, OLE etc.

PyUNO[3] Bridge – This helps the user to use the standard OpenOffice.org API from python, to develop UNO components in Python .

LibreOffice[4] 4.0 – A software package developed on the openoffice.org open source code. This is distributed with many Linux distributions.

Star Basic [5] –Star Basic provides definitions for all abstract objects and their interfaces used in programming with open office. This is used in developing macros and extensions. Similarly Visual Basic is used in developing Microsoft Office macros and extensions.

Wolfram Alpha Math Engine [6] - The aim of Wolfram Computational and Knowledge engine is to make all systematic knowledge computable and accessible to everyone. It targets three points, the first being collect and curate all objective data. The second is to implement all the known models. And algorithms and methods in the world and make it possible to compute whatever can be computed about anything as third. The developers target to make the Wolfram Math engine to be a historical milestone in the 21st century.

B. *Extensions Available vs EasyTuteLO*

The extensions developed so far seemed to be additions of new features into the word processor. The extension described in this paper integrates services into the word processor, but it outsources the computational or knowledge providing capability.

III. ARCHITECTURE OF THE EXTENSION

The architecture of the extension can be described using layered architecture and the component based architecture.

A. Layered Architecture

The extensions are structured between the LibreOffice Application and the User Interface Layer. It extends the features available for user. This can be easily identified using the Figure 1. The EasyTuteLO Project was implemented in the way the Figure 1 describes. This will be the case for most extensions for the LibreOffice. As it is shown in the Figure 1 the extensions may use the services available on the operating system. In the EasyTuteLO extension it uses services outside this system also.

B. Component Architecture

- File Handler
- Communicator
- Query Converter
- Document Handler
- Result Converter
- UI Generator

The above components can be easily identified in the component based architecture. The Figure 2 gives a better illustration about their interactions.

Communicator does the basic communication between the server and the word processor application.

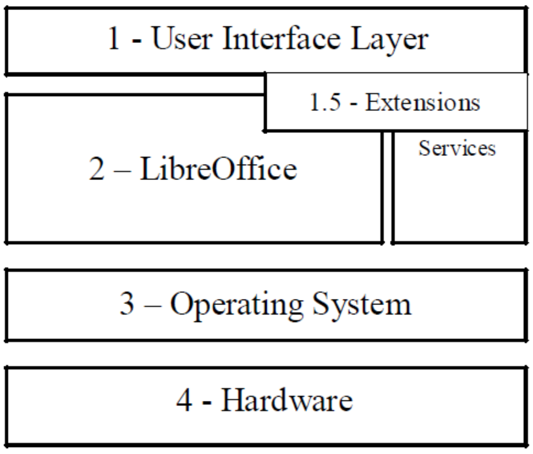

Fig. 1. *Layered Architecture of the extension*

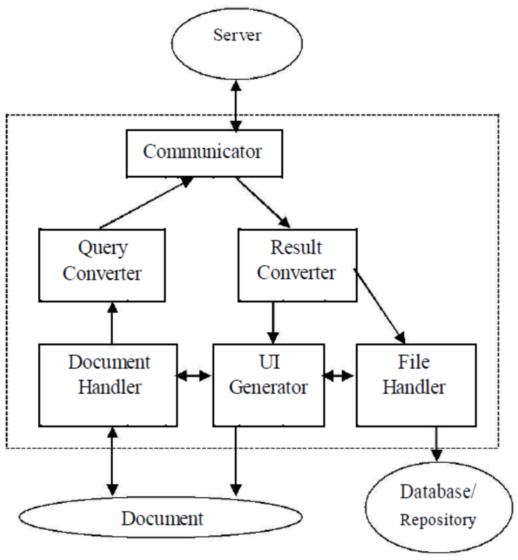

Fig. 2. *Component based Architecture of the extension*

File Handler is responsible for retrieving files from the server and persist them in defined locations and retrieve them for use when needed. Query converter converts the query typed in the document into the server accepted format. Document Handler controls the two parts where the typed data is passed to the query converter and the results are printed in the document. The result converter converts the result received from the server to the general format which is defined to coordinate with document handler. The UI generator is responsible for generating two user interfaces, the query input interface and the available results interface.

The Document Handler interacts with the document to get the query from document and pass it to the query converter or if it is an older query pass it to the UI generator. Query convertor coverts the query according to the server protocol and passes it to the communicator. The communicator pass the formatted query into the server, then it receives a result and then the result will be passed to the file handler. Simultaneously it informs the UI generator about the availability of the result. File Handler manages the two way communication between existing database and the repository. It sends the retrieved data to generate the user interface to see the availability of the data. The UI generator passes the user interactions to the document handler component and the file handler component

IV. IMPLEMENTATION

A. Description of the functionality

The described design was implemented using Python Language and the UNO API. This was named as EasyTuteLO as it was focused on building mathematical tutorials easily. In that implementation all the mathematical functionalities were not added. Several functionalities such as Integration, differentiation, differential equations etc. were implemented. In the EasyTuteLO extension two basic routines were defined. One routine is to retrieve the details for the very first time. Once the user has typed the required query, it is sent to the extension and the execution of the extension is carried out. The Wolfram Alpha Computational engine provides a web service to communicate with it for computational purposes. The retrieved data is stored in the specified locations and they will be shown in an available data window. Then the users can select whatever the data they need and those can be added to the document.

The other routine handles the previously fetched data. In that routine, the previously handled queries are displayed. The user can select one of them and see the available resources. Then the required data can be selected. The selected items will be added to the document as in the first case.

B. *Implementation Process*

The elements involved in developing the extension were following.

- Coding and debugging

Eclipse IDE was used for implementation, the testing and debugging of the code. The PyUNO Bridge needs to be installed to access open office components at the office run time. Once UNO service is started, the code can be debugged using the IDE.

- Testing

The test package python unit test was used to write unit testing for the functionality implemented. Since the coding does not need any other modification apart from the PyUNO Bridge configuration, the testing didn't take any newer approach.

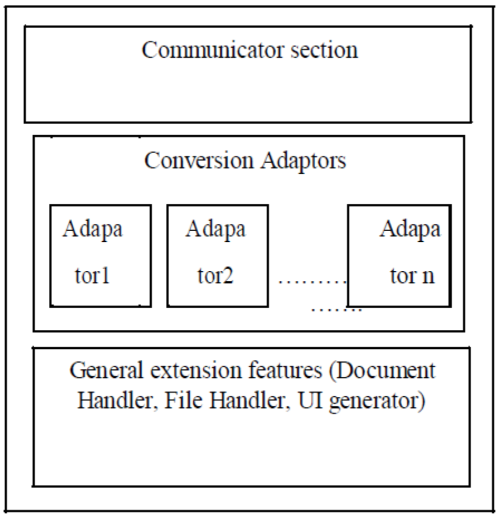
Communicator section
Conversion Adaptors
Adapa tor1
Adapa tor2
………
Adapa tor n
General extension features (Document Handler, File Handler, UI generator)

Fig. 3. *A generalized Architecture of the extension*

- Compiling

Because python is an interpreted language, compiling topic does not play a major role in the implementation process.

- Packaging

Packaging involves loading several files and packaging them into one file. Then the file will be unpacked at the extension installation. The extension details, licensing details and the LibreOffice UI modifications, libraries and other scripts are packaged here. There exists an extension packager [7] which eases the life of the developer. The extension packager creates the extension setup file once the specifications are given to the packager.

- Documentation

The documentation for the Libreoffice extension is not sufficient in learning the extension development. Many of them still refer to the Apache Open Office development tutorials. At first glance it seems somewhat confusing. Moreover the documentation for building extensions using Python was a very few number. Due to this reason the code was written by looking at the Java examples. At this point a very important observation was noted which is the python code is approximately 1/3 in length when comparing to the respective java code.

V. FUTURE WORK

The architecture of the software can be generalized in order for many services to be integrated. As the experiment describes, the Wolfram Alpha Computational Engine is capable of providing various services. In this experiment only the mathematical queries were used. Nevertheless it is capable of providing knowledge on music, astronomy, humor, news, geography, history etc. Here the mentioned fields are not relevant to each other which reflect the scope of the computational engine.

Therefore, the extension can be generalized to the architecture described in Figure 3 enabling it to be more helpful in word processor activities and multipurpose.

The adaptor libraries to deal with servers are added to the extension. The adaptor library should be consisted with two parts; one is query conversion and the other is result conversion. The target of the adaptors here is ultimately matching the interfaces. Hence the library's main task is to act as the interface between that general format and the server specific language.

VI. CONCLUSION

The word processor applications consist with various features. Currently their trend moves towards implementation as a cloud based application. In future their trend may be to give more knowledge and computational power to it. Currently the word processors are capable of doing computations but the user experience seems lower and because of that reason the user has to do the computation and information gathering separately in order to prepare the document.

A. *Evolution of word processors*

With past experience everyone knows that the word processors were gradually added new features. At the beginning, they were capable of doing very primitive operations such as editing text and storing them. But the graphic and other element support, macros and cloud based infrastructure were also integrated gradually. Currently the word processors fulfill the user needs to a certain extent where they need to go a long way. The word processor evaluation can be categorized into following according to their paradigm shifts.

1) *Basic Word Processor*
2) *Text format-able*
3) *Graphic and other element support*
4) *Macro enabled and extendable*
5) *Word Processor Software as a service (Cloud based)*
6) *Has Knowledge and Computational Power*

The text editors where only text editing and storing is supported are categorized into Basic Word Processors. The text editors which are capable of changing the properties of characters can be included into the Text format-able category. There are some word processors where graphics, images, charts and other related elements can be inserted into documents; such software come into the third category. Some word processors are capable of executing macros; they are also capable of installing extensions. Openoffice writer, Microsoft office word, Libreoffice writer word processors come into this fourth category. Most recently the cloud based word processors were implemented; such as Google docs. Those can be categorized into the 5th category. The extensions similar to the one described in this paper can be a default feature in future word processors, in such a scenario the most frequently used knowledge and computational needs can be fulfilled by connecting with servers. Such implementation can be categorized into the 6th category. If such functionality becomes a default feature of the future word processors they can be the next generation of Word Processors.

Moreover the services can be HCI (Human Computer Interaction) related services in the future. If HCI features such as voice are supported in word processors, it can be categorized into another generation.

ACKNOWLEDGMENT

The research paper was written based on the project carried out for the Semester 5 module Software Engineering Project of Department of Computer Science & Engineering at University of Moratuwa. The lecturers, instructors, evaluators and the coordinators of the project gave numerous support in succeeding with the project. The final year undergraduates who have finished this module in the previous year also gave a great help for the success of the project. A hearty gratitude goes to everyone who supported the project.

REFERENCES

[1] Python Software Foundation, Python v2.7.6 documentation, http://docs.python.org/2/

[2] The Document Foundation, Libre Office Developer Guide, http://www.libreoffice.org/developers/.

[3] Apache Open Office, PyUNO Bridge Wiki, https://wiki.openoffice.org/wiki/PyUNO_bridge

[4] The Document Foundation, Libre Office Extensions, http://extensions.libreoffice.org/

[5] Sun Micro Systems, Star Office Programmer’s Tutorial, USA, May 2000

[6] Wolfram Alpha LLC, Wolfram Aplha Web Service API Reference, http://products.wolframalpha.com/api/documentation.html.

[7] Apache Open Office, Extension Packagers Wiki, https://wiki.openoffice.org/wiki/Extensions_Packager