Project Proposal – GSoC 2014

Haskell UNO Language Binding

LibreOffice - Document Foundation

Tharindu Lakmal Muthugama
University ID - 100347U
University of Moratuwa (3rd year undergraduate)
Sri Lanka

Introduction

UNO (Universal Network Objects) project was developed to implement a Component Model like architecture for the LibreOffice. The services offered by the LibreOffice core is accessible through the Binary UNO API and the UNO Remote Protocol as well.

The communication between components can be described using the UNO IDL (Interface Definition Language). Currently various language bindings for the UNO API is available. These Bridges are available for C++, Java, Python. This Project's objective is to build a similar bridge for the Haskell Programming Language.

C++ Binding

Binary UNO can be directly accessed through the C++ Language.

Java Binding

Java language Bridges with Binary UNO via JNI

Python

Python Language Bridges with the Binary UNO via the pyuno bridge

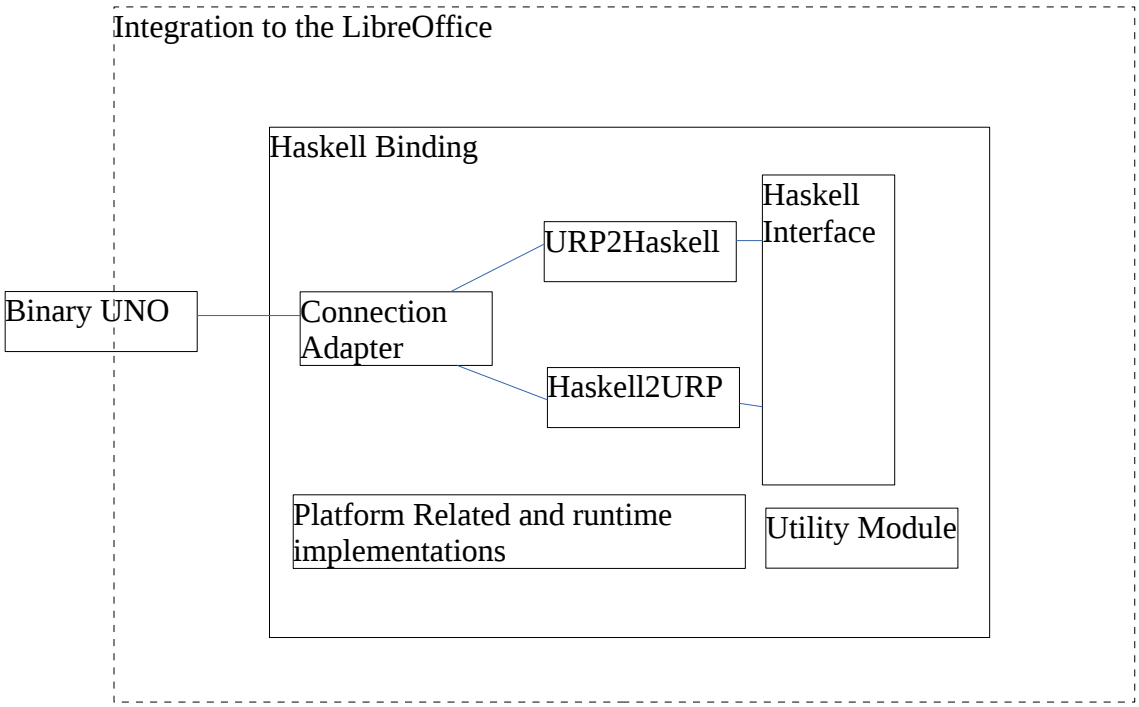
URP Bridge

URP is available to bridge with any other implementation

Proposed implementation for the Haskell

A purely Haskell implementation is proposed to handle the URP and invoke the services available.

Design Details of The Project



The Haskell binding communicates with BinaryUNO through a connection adapter. The Messages communicated via the connection adapter will be translated between Haskell Language via the modules URP2Haskell and Haskell2URP.

The Runtime implementation module will communicate with the other processes in the environment. The integration to the LibreOffice will also considered as a separate phase.

A utility Module is used to get services required in the implementation. Eg. Logging

The design, documentation and testing work will also be carried out parallel to the coding. The work break down for the project is defined below.

Tasks of The Project (Work Break Down)

1. Familiar with the UNO Model thoroughly
    - Illustrate the Java UNO Bridge using UML
    - Illustrate the python UNO Bridge using UML

2. Familiar with the Haskell language thoroughly
    - Implement a basic communication protocol

3. Familiar with the UNO RPC Bridge using UML
    - Gather details about RPC Protocol
    - Gather details about UNO IDL

4. Write the Connectivity part from Haskell
    - Access the URP Port and get replies
    - Experiment with the URP with the specifications given

5. Implement the URP Adapter from Haskell to receive messages
    - Documentation about the implementation
    - Implement
    - Develop test cases and test

6. Implement the Haskell2URP Module
    - Documentation about the implementation
    - Implement
    - Develop test cases and test

7. Implement the URP2Haskell Module
    - Documentation about the implementation
    - Implement
    - Develop test cases and test

8. Integration of Adapter and URP2Haskell and Haskell2URP Module
    - Documentation about the implementation
    - Implement
    - Develop test cases and test

9. Implement the Platform related and Runtime implementation
    - Documentation about the implementation
    - Implement
    - Develop test cases and test

10. Integration and Refactoring
    - Documentation about the implementation
    - Integration
    - Refactoring

11. Integration Testing
    - Develop Test Cases
    - Testing

12. Documentation and Develop sample projects using the language binding
    - Documentation about the implementation
    - Develop Tutorials to code with Haskell Language Binding

Utility Module will be developed throughout the entire project period since it is added the required utility methods at the point of implementation.

Time line

Period	Task	Description	Other Calendars
10th March – 21st March		Preparing The Project Proposal Get involved in the Development community Bug fixing	
21st March – 18th April	Task 1 Task 2	Bug Fixing to get the knowledge about code base. Task 1 and Two will provide thorough domain knowledge about UNO and Haskell.	
21st April – 19th May	Task 2 Task 3 Task 4	Task 2 and Task 3 will provide a better design and preparation for the project by familiarizing more to an environment where RPC, UNO and Haskell is involving.	
19th May – 26th May	Task 5	The connection adapter provides the services offered by the BinaryUNO. Which will be needed in order to work ahead. Hence it is implemented as the first module.	14th - 23rd May – End of Semester 6 (evaluation will be held within a day in that period) 26th May – Semester 7 starts
26th May – 2nd June	Task 5,6	The continuation of the Task 5. The Haskell2URP module will be implemented.	
3rd June – 9th June	Task 6	Continue the implementation of Task 6	
10th June – 16th June	Task 6,7	Continue the Task 6. Start implementing the URP2Haskell Module	
17th June – 23rd June	Task 7	Continue the URP2Haskell	
24th June – 27th June		Mid Term Evaluation and get feedbacks about the implementation and discuss about future work.	
28th June – 5th July	Task 8	Integrating the previously implemented modules and develop test cases in order to continue with a working and tested code.	19th July – 28th July (Mid semester exams will be held within the period)
6th July – 12th July	Task 8,9	Continue the Task 8 . The platform related and runtime implementations will be handled after that.	
13th July – 19th July	Task 9	The task 9 will be continued.	
20th July – 26th July	Task 10	Integration to the LibreOffice and refactoring of the code.	

27th July – 3rd July	Task 11	Integration Testing. End to End test cases will be developed.	
4th July – 10th July	Task 11,12	Continue Task 11. Some Example projects will be implemented using the developed Language Binding. This will be focused on better documentation in order to remove the obstacles for future developers and third party developers.	
11th July – 17th July	Task 12	Continue the Task 12.	
18th August – 22nd August		Final Evaluation	
22nd August – 25th August		Submitting required code samples	

Alternative Approach

Use Haskell Foreign Function Interface to invoke the services in BinaryUNO

Sample code

```
module Example1 where
import Foreign.C

foreign import ccall "cot" hask_cot :: CDouble -> CDouble
```

Above hask_cot method will be bounded to the cot method in Math.h C header file. Similarly the Haskell methods will be created to invoke a UNOHaskell Bridge similar to PyUNO.

About The Applicant

Name : Tharindu Lakmal Muthugama
Date of Birth : 3rd July 1990
University : University of Moratuwa
Melange Handle Name : tmtlakmal
IRC Nick name : tmtlakmal
E – mail : tmtlakmal@gmail.com
Linkedin : http://www.linkedin.com/pub/tharindu-lakmal-muthugama/56/441/804
Blog : http://tmtlakmal.wordpress.com/
GitHub : https://github.com/tmtlakmal

I have written engaged in coding for few years. My programming career started from the C Language. Then it was changed to Java. After several assignments in the

university I was able to develop a java application related to Google Maps. The experience I gained through the Tank Game Project with Java and the Database Project with CakePHP framework motivated me a lot into learn new technologies.

The quiz system project I built for Labnoir Pvt Ltd improved my confidence of learning new technologies. Though both MongoDB and the Dart Programming Language was new to me at that time I was able to manage and finish the project successfully.

I wrote blog posts regarding python macros in LibreOffice which I experienced while doing the EasyTute LibreOffice Project which is to get solutions from Wolfram Alpha Math Engine for the equations type in LibreOffice Writer. That became the first experience with UNO and Python Language. As I have worked with UNO API it made me easier to get the understanding about the Language Binding implementations for Java and Python.

Currently I'm fixing LibreOffice bugs , implementing several Haskell mini projects and exploring the source codes of URP, JniUNO and PyUNO in order to gain thorough knowledge for the success of this project.

References

1. https://wiki.openoffice.org/wiki/PyUNO_bridge
2. https://wiki.documentfoundation.org/Development/Gsoc/Ideas
3. http://cgit.freedesktop.org/libreoffice/core/tree/bridges/source/jni_uno?id=fa97b8ac234c34618d8dca4329bc13e8454b47b4
4. http://cgit.freedesktop.org/libreoffice/core/tree/binaryurp?id=fa97b8ac234c34618d8dca4329bc13e8454b47b4
5. http://www.openoffice.org/udk/common/man/typesystem.html
6. http://www.openoffice.org/udk/common/man/lifecycle.html
7.https://wiki.openoffice.org/wiki/Uno/Remote/Specifications/Uno_Remote_Protocol
8. http://www.openoffice.org/udk/common/man/uno_the_idea.html
9. https://wiki.openoffice.org/wiki/Uno/Article/Understanding_Uno
10. http://www.haskell.org/haskellwiki/Haskell
11. https://wiki.openoffice.org/wiki/PyUNO_samples