Macro Translation

Asher Gingerich

Department of Computer Science & Engineering • Taylor University • 236 W. Reade Ave, Upland, IN 46989

Introduction

OpenSolver is a macro based extension for Microsoft Excel. Microsoft macros are created using Visual Basic for Applications (VBA). One major obstacle in the porting process is that LibreOffice uses a different macro language: LibreOffice Basic. These two languages are similar in many ways, using similar keywords and syntax, but are just different enough to be incompatible with one another.

Further complicating the porting process is the interconnected nature of many of the functions within the OpenSolver implementation.

Diagramming

Step one in porting was to find a way to simplify the process of moving from function to function. Due to the many function dependencies throughout the codebase, the simplest way to begin implementation was to map out those dependencies. This map effectively made an ordered todo list, showing which functions were good starting points for implementation and testing.

VBA Support

The biggest impediment to porting OpenSolver is the need to convert from VBA to Basic. LibreOffice does provide some support for VBA, which speeds this process up a bit, since not everything needs to be translated or modified. The problem with this support is that it is only partial support. There are many VBA functions that do not work exactly like they are expected to, needing alternative solutions to provide that functionality.

Named Values & Ranges

Running OpenSolver relies on a feature within Excel and LibreOffice that allows the user to name and save values and cell ranges. OpenSolver uses this to save the locations of the various necessary elements to solving a linear programming problem as essentially global variables. A positive side effect of this is that these named ranges are saved to the workbook document itself, and therefore are present between sessions in the worksheet.

There are two main functions that handle this. One function handles saving named values, while the other handles saving named ranges. These are then called by wrapper functions that feed into them the appropriate inputs for various data types (for the values) or how to handle missing data (for the ranges).

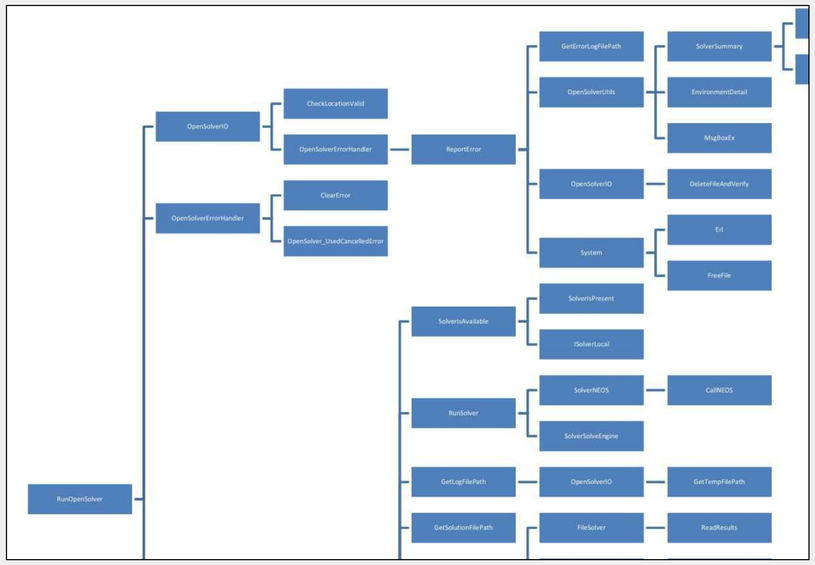

Figure 1. Section of the dependency diagram

API

Every necessary variable to use the solver has its own method of getting and setting these values, so most of the API is devoted to managing these. After type conversions and data modification, they all feed into two name setting functions that either set named values or named ranges onto the sheet. Therefore most of the API exists as easy to digest wrappers for other functions, largely with similar functionality and function calls.

Conclusion

During my time with this project, I learned the importance of documentation as I wrestled with the lack of up-to-date, in-depth documentation for LibreOffice Basic.

While much of the extension still needs to be implemented and tested, the completion of the named range handling the API, and many utility functions allows future developers to focus on the higher level functions and translations.

Future Work

There is much that still needs to be done with the translation to reach a full port. Sensitivity Analysis still needs to be handled, and the output sheet created. There also needs to be continued work with the many different solver types that OpenSolver uses, and the many class modules that it relies on.