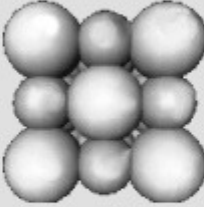


Joseph Moosman		www.ALATI.se	
Applications		IT Solutions	
Alstigen 5		Tel: +46 (0)505 - 109 25	
S-546 30 Karlsborg		Cell: +46 (0)73 - 637 80 11	
SWEDEN		jmoosman@gmail.com	

User-Safe Excel Application Development

Microsoft Excel is one of the most widely deployed and used computer applications on the world market today. There is a large-scale, international market for the development and implementation of complex, end-user systems written in Microsoft Excel. These systems are designed to facilitate and support data processing and modeling tasks which are too complex and/or extensive for manual completion in Excel.

Excel system development is supported by Excel's internal programming language, VBA (Visual Basic for Applications) and through the use of complex formulas written in spreadsheet cells using Excel's formula syntax.

A very common problem that arises in the deployment and real world use of Excel applications is the inherent vulnerability of the Worksheet (spreadsheet) object model. Simply stated, spreadsheet pages are highly susceptible to damage through end user intervention. Users inadvertently (or deliberately) edit formulas or break up data lists and spreadsheet Ranges that must be left intact for the intended model to function reliably.

Locking down XLS format spreadsheets with Excel's Protect and Lock method sets is, at best, a partial solution to the fragility of complex sheets: this level of protection can be easily overridden by end users and does not stop them from saving the Workbook object in the form of saved XLS files that can represent unlocked, incomplete or erroneous states of the application model. This last vulnerability is particularly serious when there is a demand for a reliable audit of Excel calculations that are to be used in important decision making processes. A proliferation of Workbook versions containing different calculation states

makes this very difficult and is a common failing of enterprise scale Excel solutions.

Another method frequently used to address this problem is the packaging of complex Excel solutions in Excel's XLA add-in format. XLA add-ins are "compiled" representations of Excel Workbooks which contain all the VBA and spreadsheet logic of the original XLS document. XLA add-ins are almost always used to deliver "user defined functions" (UDFs) which are VBA functions that can be accessed from the spreadsheet in the form of formula elements. Another very common XLA technique is the implementation of additions and changes to Excel's own menus. These new menu items are used to trigger UDF calculations on spreadsheet cells. UserForms are also deployed through XLA packages. UserForms represent "user friendly" dialogs that also trigger UDFs and concomitant calculations on spreadsheet cells.

This type of XLA deployment suffers from all the liabilities of the Locked XLS method described above. All operations are carried out on the user's Worksheet and can easily be corrupted or spread throughout an organization in contradictory versions. Also, XLAs in their typical forms increase end user uncertainty by hiding important application choices and methods in otherwise unfamiliar or modified Excel menus. UserForms are static, modal dialogs which actually limit communication between the user and the application. They lack the interactivity of the Worksheet model and are also hampered by an unimaginative and outdated `.Load`, `.Show`, `.Hide` and `.Unload` method set.

What is missing in these established methods is a way to combine the flexibility of the Worksheet model with the safety and reproducibility of the XLA package. This is exactly what I have done in my application development method.

My method has the following feature set:

- 1) No data, including user preferences and application state, is saved or stored in any user accessible Excel format. All data are saved in relational databases. The usual database choice for desktop applications is Microsoft Access which I implement through the `CreateDatabase` method. This means that the target machine does not have to have Access installed. Data can also be stored in Oracle tables, MySQL or any other available relational database system.

- 2) Interactivity is preserved in the XLA compiled application through a complete encapsulation of the XLS Worksheet object model in the XLA document. This means that users interact with Worksheets that are entirely generated and controlled by the XLA logic and that they can not save any changes in these files. The users can only interact with the application on customer specified

terms - but they still have all the layout and calculation features of the Worksheet model at their disposal.

3) The application is always in the same state when it is started and this makes possible a far simpler audit and verification of both results and ensuing decisions.

4) Users can save "snapshots" of their models to XLS documents. These files are completely independent of the application that created them - there are no external links or references back to the XLA application. The saved documents are, however, carefully and thoroughly labeled, both externally and internally, with information on all the factors, including version number and database status, that were in force when they were created. This means that the parameters that were used to create any particular model can easily be inputted back into the application for validation or modification. It also means that users can comment, illustrate, modify and share their own viewpoints on the models - without affecting other users' work or the stability of the application.

5) Because of their robust construction, my applications can be easily shared on network servers to many simultaneous users. All users have access to the same copy of the same application and can not save any conflicting versions. This means that maintenance is far easier than trying to support and audit an unknown number of "competing" Excel Workbook versions of the same model. Upgrades consist of simply overwriting the XLA application file with a newer version.

6) Because data is stored entirely separately from application logic, upgrades can easily be carried out without overwriting saved user data. This task is extremely difficult to accomplish in Excel applications in which data is stored in the Excel XLS workbook.

7) Because all logic is encapsulated in the XLA document, Excel's default user interface can be modified at start-up and restored at shut-down. This means that any and all Excel menus and user functions can be selectively hidden by the application to avoid any attempt at unauthorized changes to the working model. This can be taken so far that the text "Excel", for example, is completely missing from all application screens and even experienced users are not certain that they are using an Excel application. Users at all levels of Excel proficiency, from novices to experts, can use the application effectively without having to dig down into the inner workings of the models being deployed.