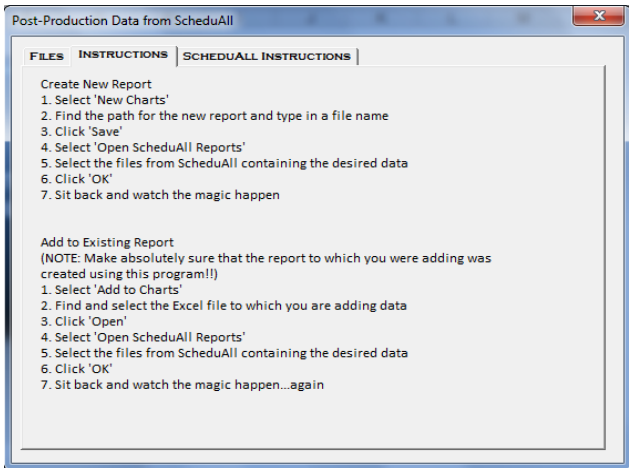
Karl Rosengren
April 13, 2010

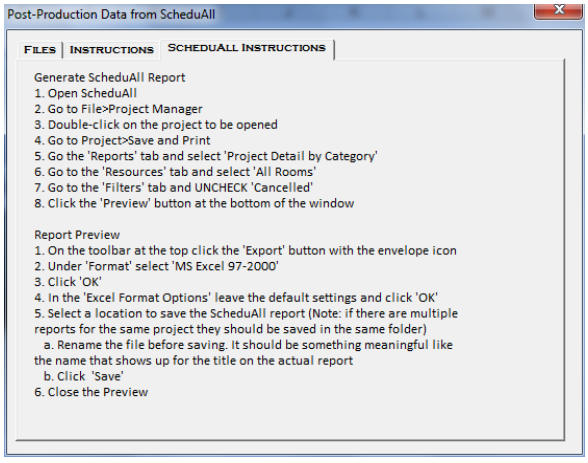
The goal of my project was to automate a process taking data from another piece of software called ScheduAll and importing it into Excel in a format that can be easily manipulated and analyzed. ScheduAll actually exports to an Excel format but it is a format that is meant more for printing than it is for analysis so it needs to be translated before it will be useful. Please note, this issue came up because of a part time job I had requiring this analysis. The translation process, by hand, could take upwards of two hours or more depending on how much data a project contained.

The initial user interface is essentially a giant button which, when pressed, will call a sub procedure which launches a user form. The user form, when first initialized will have a page with three tabs. The second and third tabs load with text describing the process for getting the appropriate data out of ScheduAll and the process for using the current user form. The first tab has three buttons which will each launch another sub procedure when clicked. They are dormant, otherwise.

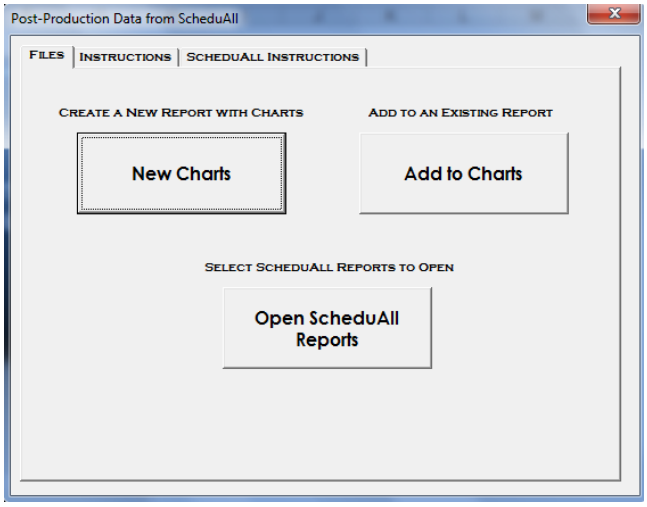
Screenshot 1: Initial User Interface



Screenshot 3: Report Translation Instructions



Screenshot 2: ScheduAll Instructions



Screenshot 4: User Form Buttons

Pushing the "New Charts" button will call a sub procedure called "MakeSaveFile." This procedure uses the GetSaveAsFilename dialog box to allow the user to specify a file path and file name. Essentially, the code allows the user to save a new Excel file wherever he chooses. The file name is stored as a string where it can be recalled later in the process. After the path is saved a new workbook is created and saved using the specified path and file name. This file will be the new home of the translated data where it can later be used for our analysis. The code also does some formatting to the new Excel file such as deleting extraneous sheets and renaming the one sheet that it does keep. That sheet is renamed "DestinationSheet." Across the top row some column labels are added for use later, as well.

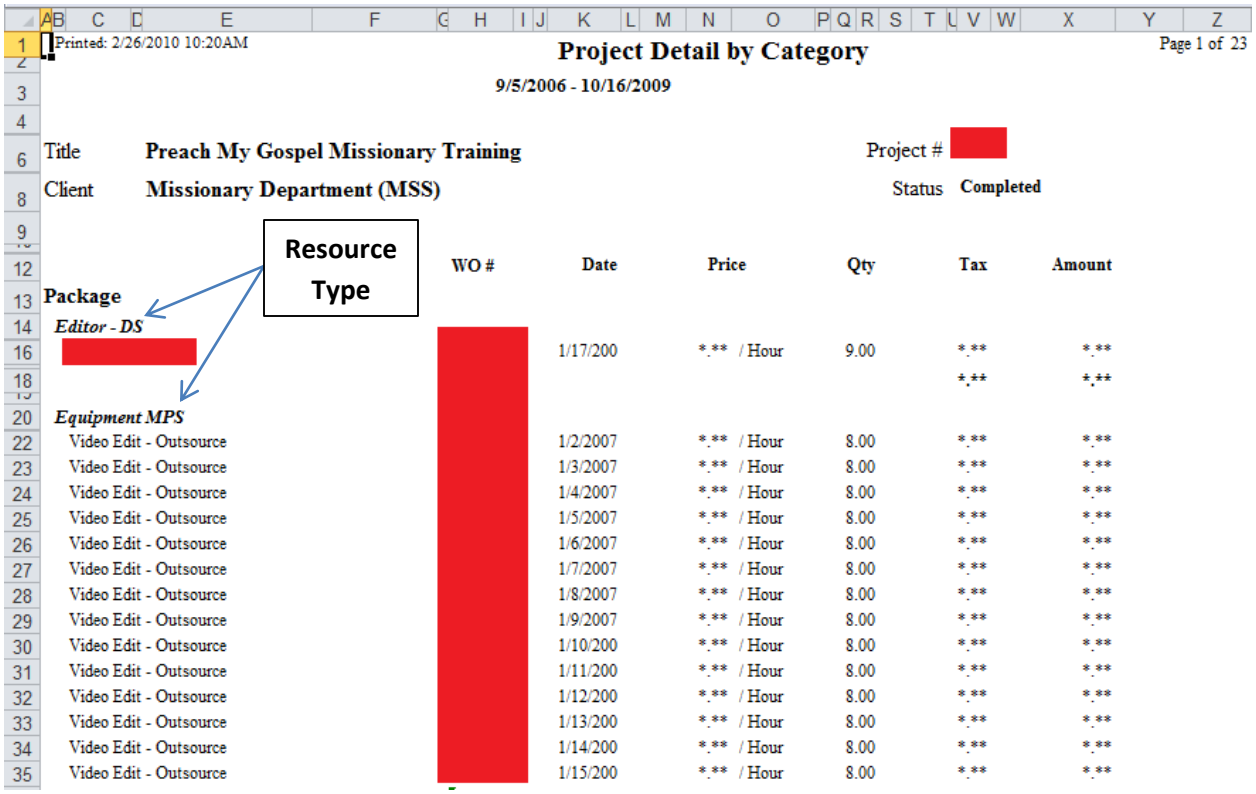
Note: Throughout this process there is a check value used in case the user clicks the "Cancel" button in any dialogue box. Depending on what is happening, clicking "Cancel" may either take them back to the user form or it may cancel them out of the code altogether.

The creation of the new workbook has happened in the background and it is now visible behind the user form. The next step would be to click on the "Open ScheduAll Reports" button. This calls a sub procedure called "OpenSheet." This procedure opens a FileDialog dialog box to allow the user to choose the ScheduAll reports he wants to import into Excel. He can select multiple files but they must all be located in the same folder. The code will look at each file one at a time and go through the "CopyData" sub procedure. More on that later. After the code goes through each selected file, the code sets the open file to nothing and ends the procedure, returning the user to the code in the user form.

Let's go back to the CopyData procedure. After setting needed variables, the code starts at the top of the active sheet (which is the "DestinationSheet" specified above) and increments down until it find the first blank row. That is set as the destination row for the first line of copied data. Then the first of the selected files in the OpenSheet procedure is actually opened and set as the active workbook. Then, based on the specific layout of this report, we do a search for "Grand Total" in the cells. In column C in the same row as "Grand Total" a mark is put in to let the code know when it has reached the end of the data in this file. Then certain string variables are defined as data from this report that will be common to every entry in the report, such as the project title and the client.

Column C is used to find a new entry. So we start at the top of the column using multiple "do" loops to transfer the data from this sheet to the new workbook we have created. One interesting thing about this data is that it may be of a certain "type." This is shown on the report offset up and to the left of the

entry description. An offset is used to check for a value. If there is a value then it is stored as a string to be copied as well. If not then we get to the loop that actually transfers the data. This loop is defined by cycling through until the row in column C is empty. Then it skips down in that column until it finds



Screenshot 5: ScheduAll Sample Data

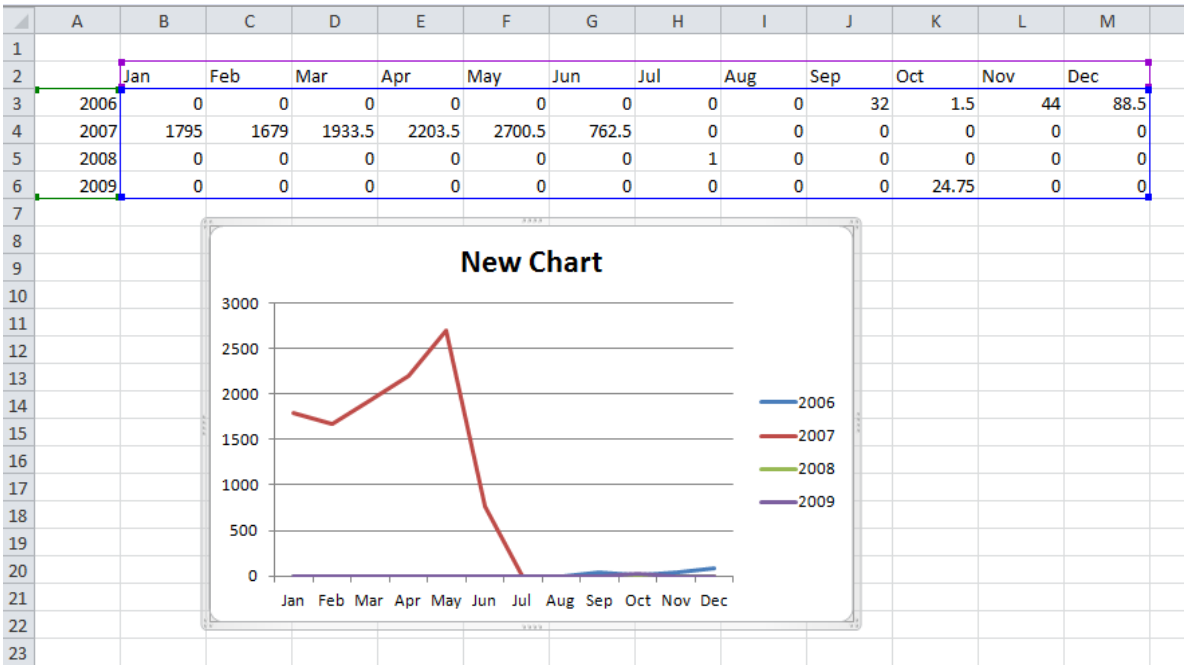
another value. If that value is the same as the mark we placed earlier, then we know we have reached the end of the data. To copy the data, the value for the cells in the destination workbook are set equal to either the strings defined earlier or to the cell values shown in the data.

Once the end of the data is reached, the mark is cleared at the end of the sheet and the file is closed with savechanges set to false. Then the destination workbook is set to be active and the procedure ends, taking us back to the OpenSheet procedure which will go to the next file or go back to the user form.

Once all of the ScheduAll reports are copied, the user form code associated with the "Open ScheduAll Reports" button then calls another sub called "Pivot." This procedure creates a new sheet in the destination workbook in which to create a pivot table based on the data in the already existing sheet. If, for some reason, this sheet already exists, it will be deleted and created again. Then the parameters for the pivot table are entered and it is created. In addition, certain fields are preselected to be shown in the table. The resources are shown by type and collapsed to just show the type as the row headers and the date is the column header. The date data is grouped to show as year and month. The main data shown is the sum of hours scheduled. After the table is formatted, another sub procedure called "FilterItems" is called.

Since the items are grouped by type, we want to show only the types that have the word “Rooms” in them. We do this by creating a string array which is filled with the name of each resource type. We fill the array by going down the table but we take items off from the bottom up (that way we can decrement the row number and even if the item was removed we will always be on the next row to check). An “instr” function is used to look in each string in the array and search for the word “Rooms.” If the word is found, the function will return a number greater than zero so we move on. If the word is not found, the function returns a zero and the pivot field’s item visibility is set to false. When it is done, the pivot table is sufficiently filtered and we go back to the pivot table creation procedure, which is also, done. So we go back to the user form code.

The next step in the code is to go to the sub procedure “CreateChart.” This code finds the data range for the pivot table and creates a new worksheet to copy it to. Then it copies the data going across the row (it’s actually copying data from three rows but two of them are headers). The headers are dates and once it gets to January it moves down a row on the destination sheet so the data is showing in consolidated table format. We are only interested in the Grand Total at this point so that is the only data copied. Once all of the data is copied, it is selected and used to create a new chart which is then formatted and placed close to the top of the page.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2006	0	0	0	0	0	0	0	0	32	1.5	44	88.5
2007	1795	1679	1933.5	2203.5	2700.5	762.5	0	0	0	0	0	0
2008	0	0	0	0	0	0	1	0	0	0	0	0
2009	0	0	0	0	0	0	0	0	0	24.75	0	0

Screenshot 6: Data and Chart

At this point the code is done. The user form gets hidden and close and all of the sub procedures have ended. The user is left looking at a chart of the data and can do any further analysis he desires.

The other button on the user form is only used for adding data to a file that has already been created. The process is essentially the same but it calls its own sub procedure to open the file directly and add the data based on the files selected, just like normal. The biggest difference is that this has some reminders about using it responsibly to not destroy any desired data.