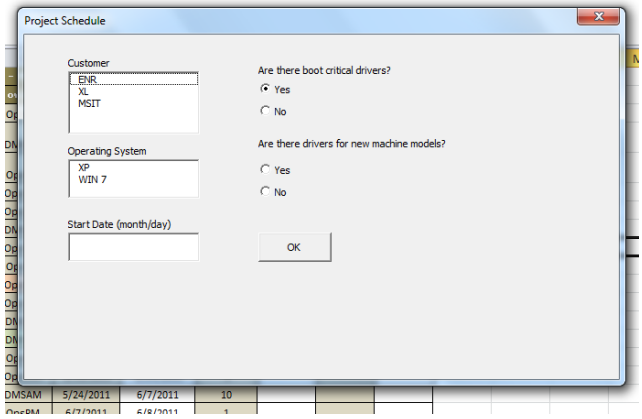
Aaron Johnson
MBA 614 Final Project

Executive Summary:

My brother is an engineer at Microsoft and he asked me to create a program to help him schedule projects. This program creates a schedule based on input from the user on the customer, the operating system, drivers, and the start date of the projects. With this information, the program creates a schedule with all of the steps necessary for that project, as well as the projected start and end date for each step. Once a project has started, if a particular step needs more time, then there is a button to enter in the new end date and each of the following steps will be adjusted accordingly. When a new schedule is being created, the program checks to see if an old one is present, and if one is, the old schedule is stored on a separate worksheet. A separate button allows the user to run a report that checks these past schedules for how many steps and projects were finished late or early.

Implementation:

The first step in completing this project was to create a user form to collect the inputs for the user. This form is shown to the user when the "New Schedule" button is clicked. I used list boxes to allow the user to select the customer and operating system. A text box was used to collect the date. The label for the text box instructs the user to enter in the month and day, but if the user also included the year it wouldn't be a problem. I used option buttons for selecting yes or no to boot critical drivers and drivers for new machines. I did this because boot critical drivers and drivers for new machine models cannot exist on the same project. See below for a screen shot of the user form as it appears in the spreadsheet.

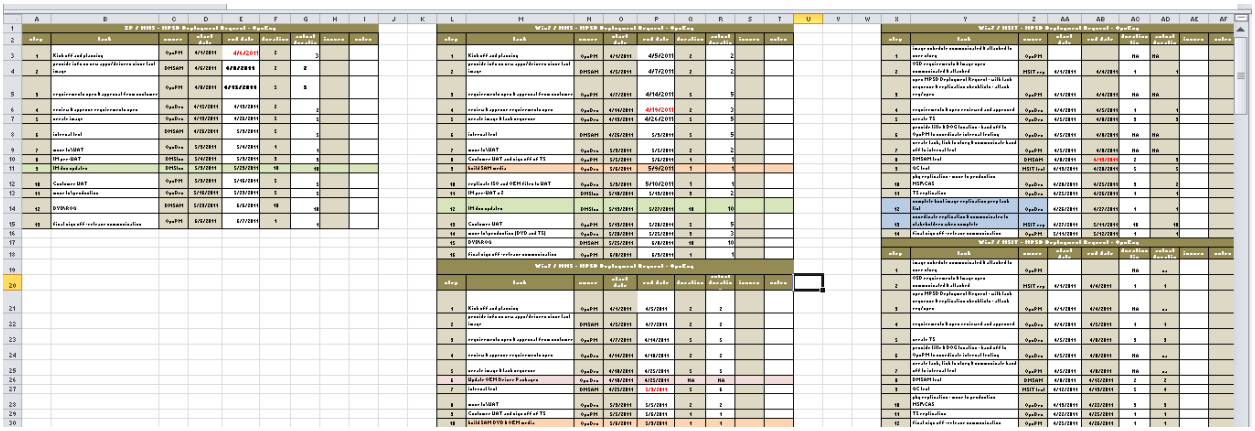


When the user presses the OK button on the user form, the program first checks to see if there is a completed schedule in the "Schedule" worksheet. This is done by checking if there is a value in the actual duration column for the final step (highlighted in yellow below).

	A	B	C	D	E	F	G	H	I
1	Win7 / MMS – MPSD Deployment Request – OpsEng								
2	step	task	owner	start date	end date	duration	actual duration	issues	notes
3	1	Kick off and planning	OpsPM	4/1/2011	4/5/2011	2			
4	2	provide info on new apps/drivers since last image	DMSAM	4/5/2011	4/7/2011	2			
5	3	requirements spec & approval from customer	OpsPM	4/7/2011	4/14/2011	5			
6	4	review & approve requirements spec	OpsDev	4/14/2011	4/18/2011	2			
7	5	create image & task sequence	OpsDev	4/18/2011	4/25/2011	5			
8	6	internal test	DMSAM	4/25/2011	5/2/2011	5			
9	7	move to \UAT	OpsDev	5/2/2011	5/4/2011	2			
10	8	Customer UAT and sign off of TS	OpsPM	5/4/2011	5/5/2011	1			
11	9	build SAM media	OpsDev	5/5/2011	5/6/2011	1			
12	10	replicate ISO and OEM files to UAT	OpsDev	5/6/2011	5/9/2011	1			
13	11	IM pre-UAT x 2	DMSInc	5/9/2011	5/12/2011	3			
14	12	IM doc updates	DMSInc	5/12/2011	5/26/2011	10			
15	13	Customer UAT	OpsPM	5/12/2011	5/19/2011	5			
16	14	move to \production (DVD and TS)	OpsDev	5/19/2011	5/24/2011	3			
17	15	DVD\ROQ	DMSAM	5/24/2011	6/7/2011	10			
18	16	final sign off -release communication	OpsPM	6/7/2011	6/8/2011	1			
19									
20									
21									
22									

Schedule | templates | historical

If the value is null then the historical data sub procedure is exited, otherwise a do loop is used to count how many rows there are until column A has a null value. All schedules have the same number of columns so there is no need to count that. The program then copies the exact range of the schedule, pastes it into the "historical" worksheet, and deletes that range on the "Schedule" worksheet. When pasting an old schedule to the historical tab, there are three categories that the schedule can be placed in. This is determined by the value that was in cell A1 on the worksheet. Once the category is determined another do loop is performed to find the first available location with a null value within that category, and that is where the schedule is pasted. Each category has its own do loop which is activated by a central if else statement based on the category determined by cell A1. This is because the columns remain constant but are unique for each category. I realize now that one do loop would have been sufficient had I created a variable for the column and assigned values to the column variable within the if else statement. This screenshot shows the "historical" worksheet.



The next step is to create the new schedule. There are five different schedules that can be created based on what inputs the user gave. Its only five because the XP project steps for ENR and XL are exactly the same and drivers do not effect it, they are also the same with Win 7 if there are no drivers for new machine models, critical boot drivers only applies to MSIT and MSIT doesn't use XP. Therefore the three main categories are XP-ENR/XL, Win 7-ENR/XL, and Win 7 –MSIT. A sub procedure is run that copies the correct template from the "templates" worksheet based on the customer and operating system chosen

in the user form. Once pasted to the "Schedule" worksheet, the rows are adjusted based on the customer and whether the user selected yes or no to the questions about drivers. For example, in the template shown below the row for step 6 is deleted if the user selects no for the new machine drivers, and the label for step ten is changed if the customer is XP. This is all done through simple if then statements that look at the values selected on the user form.

	A	B	C	D	E	F	G	H	I
18									
19	Win7 / MMS – MPSD Deployment Request – OpsEng								
20	step	task	owner	start date	end date	duration	actual duration	issues	notes
21	1	Kick off and planning	OpsPM			2			
22	2	provide info on new apps/drivers since last image	DMSAM			2			
23	3	requirements spec & approval from customer	OpsPM			5			
24	4	review & approve requirements spec	OpsDev			2			
25	5	create image & task sequence	OpsDev			5			
26	6	Update OEM Driver Packages	Ops Dev			NA			
27	7	internal test	DMSAM			5			
28	8	move to \UAT	OpsDev			2			
29	9	Customer UAT and sign off of TS	OpsPM			1			
30	10	build SAM DVD & OEM media	OpsDev			1			
31	11	replicate ISO and OEM files to UAT	OpsDev			1			
32	12	IM pre-UAT x 2	DMSInc			3			
33	13	IM doc updates	DMSInc			10			
34	14	Customer UAT	OpsPM			5			
35	15	move to \production (DVD and TS)	OpsDev			3			
36	16	DVD\ROQ	DMSAM			10			
37	17	final sign off -release communication	OpsPM			1			
38									

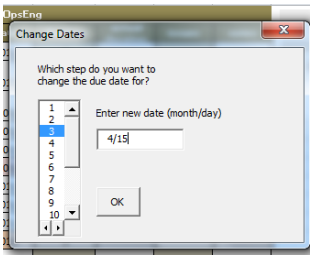
With the correct schedule in place, the start and end dates are populated. I created a module level array to store the values for the duration of each step. A new sub procedure is called that uses an if else statement to load the appropriate duration values into the array based on what inputs were given in the user form, and assign a module level variable to hold the number of steps needed in that project.

Another sub procedure then uses this array to put dates into the worksheet. The procedure starts with the date given by the user and puts that into the start date for step 1. Using the worksheet function called workday, it adds the duration stored in the first location of the array to the date given by the user and puts this value into the end date column for step 1. The workday function is useful because it automatically skips Saturday and Sunday in the calculation. This process is repeated in a for loop that ends at the number of steps stored in the module level variable that was set previously, the only difference being that the start date is simply taken from the end date of the step above it. In certain schedules this becomes a little more complicated. The following screen shot is an example of this.

	A	B	C	D	E	F	G	H	I
1	Win7 / MMS – MPSD Deployment Request – OpsEng								
2	step	task	owner	start date	end date	duration	actual duration	issues	notes
3	1	Kick off and planning	OpsPM	4/1/2011	4/5/2011	2			
4	2	provide info on new apps/drivers since last image	DMSAM	4/5/2011	4/7/2011	2			
5	3	requirements spec & approval from customer	OpsPM	4/7/2011	4/14/2011	5			
6	4	review & approve requirements spec	OpsDev	4/14/2011	4/18/2011	2			
7	5	create image & task sequence	OpsDev	4/18/2011	4/25/2011	5			
8	6	Update OEM Driver Packages	Ops Dev	4/18/2011	4/25/2011	NA			
9	7	internal test	DMSAM	4/25/2011	5/2/2011	5			
10	8	move to \UAT	OpsDev	5/2/2011	5/4/2011	2			
11	9	Customer UAT and sign off of TS	OpsPM	5/4/2011	5/5/2011	1			
12	10	build SAM DVD & OEM media	OpsDev	5/5/2011	5/6/2011	1			
13	11	replicate ISO and OEM files to UAT	OpsDev	5/6/2011	5/9/2011	1			
14	12	IM pre-UAT x 2	DMSInc	5/9/2011	5/12/2011	3			
15	13	IM doc updates	DMSInc	5/12/2011	5/26/2011	10			
16	14	Customer UAT	OpsPM	5/12/2011	5/19/2011	5			
17	15	move to \production (DVD and TS)	OpsDev	5/19/2011	5/24/2011	3			
18	16	DVD\ROQ	DMSAM	5/24/2011	6/7/2011	10			
19	17	final sign off -release communication	OpsPM	6/7/2011	6/8/2011	1			
20									

In step 7 the duration is labeled “NA”, this means that it happens at the same time and duration as the step above it, so an if then statement was needed within the loop to adjust the start date to be taken from the start date of the previous step instead of the end date. The duration was set so that the end date also matched the previous step. With step 13, all of the following steps could happen concurrently and were not dependent on the completion of this step. Therefore another if then statement was used to tell the procedure to add step 14’s duration on to the end date of step twelve.

After the schedule is created, the user has the ability to change the end date for any step in the schedule. This would be necessary if that step was either going to finish earlier or later than expected. This is done through a user form that is activated by a button on the “Schedule” worksheet. The form asks the user to select the step to be changed and enter in the new end date. The following screenshot shows this user form.



When the user clicks the OK button, the end date for the selected step is replaced with the date entered in by the user, and the font for that cell is changed to red to highlight the change. A sub procedure then runs through a similar process described above where the durations for that project are entered into an array, and the workday function adds those durations to calculate the new end date through a do loop. The number selected from the list box is used to set where in the array the workday function starts adding to the end date entered in by the user. This procedure also contains the same if then statements necessary to handle the special cases described above. The screenshots below show before and after a date has been changed.

	A	B	C	D	E	F	G	H	I
1	Win7 / MMS – MPSD Deployment Request – OpsEng								
2	step	task	owner	start date	end date	duration	actual duration	issues	notes
3	1	Kick off and planning	OpsPM	4/1/2011	4/5/2011	2			
4	2	provide info on new apps/drivers since last image	DMSAM	4/5/2011	4/7/2011	2			
5	3	requirements spec & approval from customer	OpsPM	4/7/2011	4/14/2011	5			
6	4	review & approve requirements spec	OpsDev	4/14/2011	4/18/2011	2			
7	5	create image & task sequence	OpsDev	4/18/2011	4/25/2011	5			
8	6	Update OEM Driver Packages	Ops Dev	4/18/2011	4/25/2011	NA			
9	7	internal test	DMSAM	4/25/2011	5/2/2011	5			
10	8	move to \UAT	OpsDev	5/2/2011	5/4/2011	2			
11	9	Customer UAT and sign off of TS	OpsPM	5/4/2011	5/5/2011	1			
12	10	build SAM DVD & OEM media	OpsDev	5/5/2011	5/6/2011	1			
13	11	replicate ISO and OEM files to UAT	OpsDev	5/6/2011	5/9/2011	1			
14	12	IM pre-UAT x 2	DMSInc	5/9/2011	5/12/2011	3			
15	13	IM doc updates	DMSInc	5/12/2011	5/26/2011	10			
16	14	Customer UAT	OpsPM	5/12/2011	5/19/2011	5			
17	15	move to \production (DVD and TS)	OpsDev	5/19/2011	5/24/2011	3			
18	16	DVD\ROQ	DMSAM	5/24/2011	6/7/2011	10			
19	17	final sign off -release communication	OpsPM	6/7/2011	6/8/2011	1			
20									

	A	B	C	D	E	F	G	H	I
1	Win7 / MMS – MPSD Deployment Request – OpsEng								
2	step	task	owner	start date	end date	duration	actual duration	issues	notes
3	1	Kick off and planning	OpsPM	4/1/2011	4/5/2011	2			
4	2	provide info on new apps/drivers since last image	DMSAM	4/5/2011	4/7/2011	2			
5	3	requirements spec & approval from customer	OpsPM	4/7/2011	4/15/2011	5			
6	4	review & approve requirements spec	OpsDev	4/15/2011	4/19/2011	2			
7	5	create image & task sequence	OpsDev	4/19/2011	4/26/2011	5			
8	6	Update OEM Driver Packages	Ops Dev	4/19/2011	4/26/2011	NA			
9	7	internal test	DMSAM	4/26/2011	5/3/2011	5			
10	8	move to \UAT	OpsDev	5/3/2011	5/5/2011	2			
11	9	Customer UAT and sign off of TS	OpsPM	5/5/2011	5/6/2011	1			
12	10	build SAM DVD & OEM media	OpsDev	5/6/2011	5/9/2011	1			
13	11	replicate ISO and OEM files to UAT	OpsDev	5/9/2011	5/10/2011	1			
14	12	IM pre-UAT x 2	DMSInc	5/10/2011	5/13/2011	3			
15	13	IM doc updates	DMSInc	5/13/2011	5/27/2011	10			
16	14	Customer UAT	OpsPM	5/13/2011	5/20/2011	5			
17	15	move to \production (DVD and TS)	OpsDev	5/20/2011	5/25/2011	3			
18	16	DVD\ROQ	DMSAM	5/25/2011	6/8/2011	10			
19	17	final sign off -release communication	OpsPM	6/8/2011	6/9/2011	1			

Once a project is completed, the user has the ability to call a sub-procedure that will fill-in the “actual duration” column. This is the actual number of days that it took to complete the step based on the start and end date. In most cases it will be the same as the scheduled duration in the column that precedes it. The only time that they will be different is if the user has changed the end date of that step.

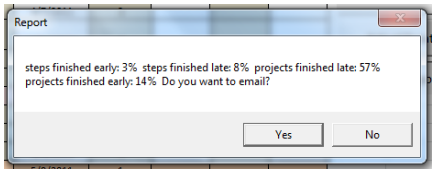
When the “actual duration” button is clicked, a sub-procedure is called that runs a loop until a cell in column A has a null value. Within this loop, an if then statement checks if the value in column A for that row is a numeric value. If this is true then, it is a step in the schedule and not a column header. It also checks the values in column F to see if it says “NA”. If this is the case, there is no actual duration to be calculated and the loop moves to the next row. If it is a row that is a step in the schedule and has a scheduled duration, another loop is used to determine the actual duration for that step. This loop runs until the workday function value for the end date plus zero equals the workday function value for the start date plus a variable that is incremented by one each time through the loop, or WorksheetFunction.WorkDay(DateValue(Cells(x, 5).Value), 0) = WorksheetFunction.WorkDay(DateValue(Cells(x, 4).Value), y) When the loop ends, the value that is in that variable represents the actual duration and is written to the cell in column G. A screen shot of the result of this procedure is shown below.

	A	B	C	D	E	F	G	H	I
	Win7 / MMS – MPSD Deployment Request – OpsEng								
	step	task	owner	start date	end date	duration	actual duration	issues	notes
	1	Kick off and planning	OpsPM	4/1/2011	4/5/2011	2	2		
	2	provide info on new apps/drivers since last image	DMSAM	4/5/2011	4/7/2011	2	2		
	3	requirements spec & approval from customer	OpsPM	4/7/2011	4/15/2011	5	6		
	4	review & approve requirements spec	OpsDev	4/15/2011	4/19/2011	2	2		
	5	create image & task sequence	OpsDev	4/19/2011	4/26/2011	5	5		
	6	Update OEM Driver Packages	Ops Dev	4/19/2011	4/26/2011	NA			
	7	internal test	DMSAM	4/26/2011	5/3/2011	5	5		
	8	move to \UAT	OpsDev	5/3/2011	5/5/2011	2	2		
	9	Customer UAT and sign off of TS	OpsPM	5/5/2011	5/6/2011	1	1		
	10	build SAM DVD & OEM media	OpsDev	5/6/2011	5/9/2011	1	1		
	11	replicate ISO and OEM files to UAT	OpsDev	5/9/2011	5/10/2011	1	1		
	12	IM pre-UAT x 2	DMSInc	5/10/2011	5/13/2011	3	3		
	13	IM doc updates	DMSInc	5/13/2011	5/27/2011	10	10		
	14	Customer UAT	OpsPM	5/13/2011	5/20/2011	5	5		
	15	move to \production (DVD and TS)	OpsDev	5/20/2011	5/25/2011	3	3		
	16	DVD\ROQ	DMSAM	5/25/2011	6/8/2011	10	10		
	17	final sign off -release communication	OpsPM	6/8/2011	6/9/2011	1	1		

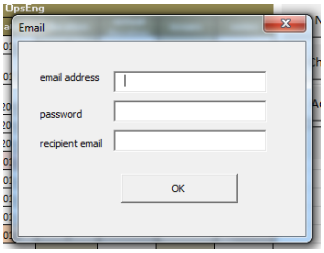
The last button on the “Spreadsheets” worksheet allows the user to run a report on past projects stored in the “historical” worksheet. When this button is clicked, a sub procedure is called that runs through a series of loops. The first loop runs until the first column of the first category of schedules is a null value. Within this loop there is a series of if then statements. The first one adds 1 to a counting variable if the

current row is a step that does not contain “NA” in the duration column. This variable is used to total all of the steps in the “historical” worksheet that have scheduled durations. The next if then statement looks to see if the current row is the last step in a schedule. If it is, then 1 is added to a counting variable that keeps track of the number of projects in the “historical” worksheet. It also adds the duration for each step in the project and compares the total scheduled duration to the total actual duration. When scheduled duration is greater, a 1 is added to a variable that stores the number of projects that finished early. When actual duration is greater, a 1 is added to a variable that stores the number of projects that finished late. If the current step has a duration value that is not “NA”, then a comparison is made between the actual duration and the scheduled duration for that step. When actual duration is greater, a 1 is added to a variable that stores the number of late steps. When the scheduled duration is greater, a 1 is added to a variable that stores the number of early steps. This same process is repeated in a loop for the second category, and a loop for the third category.

Next the number of early steps is divided by the total steps and multiplied by 100 to get the percent of early steps. The same is done for late steps. The number of late projects is then divided by the total number of projects and multiplied by 100. Again, the same is done for early projects. These results are then displayed to the user in a message box with the option to send them in an email. This message box is displayed below.



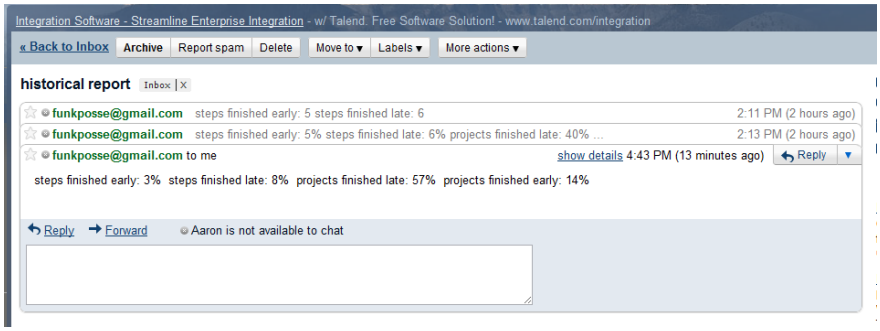
If the user clicks on yes, a user form comes up to allow them to enter in the email address that will be used to send the report (it must be a Gmail account), the password for that email address, and the email address of the recipient.



The values from each text box are stored in variables and used as parameters that are passed to a function that sends the email. This function takes these variables along with a subject string, and the body of the email, which is the information displayed in the message box. The information from the message box is stored in a string variable, and this is what was used to pass the information to the function.

The function then uses CDO to send the email out from a Gmail account. It will only work with Gmail as this is the server that it is setup to use. The subject will always be “historical report” but the body of the

email changes based on the results of the report. The screen shot below is an example of one of these emails.



This concludes the functionality of my project.

Learning and Conceptual Difficulties: One of the things that I learned was to have a good understanding of what each piece of code was doing and to not use guess and check while problem solving. Within my code I had several loops with multiple variables that were changing each time through the loop. The first time that I wrote one of these loops, it was inevitable that something would go wrong. Either the values were being placed in the wrong cells, the wrong values of an array were being added to the wrong date, or the loop was lasting too long. The first couple of times that this happened I simply tried to change one number and run the program again. This usually resulted in a lot of frustration and not many positive results. It wasn't until I started closely looking at each line and truly understanding what was happening that I would find my error.

This also helped me add a function that I thought I wouldn't be able to do at first. Creating the procedure to automatically calculate the actual duration wasn't in my original plan. It was something that was going to be added in by hand by the user, but when I tried doing this myself I realized it was pretty cumbersome. I knew that there was a datediff function in VBA that would calculate the number of days between two dates. So, I thought it would simply be a matter of taking the difference between the start and end date. However, this function failed to account for Saturdays and Sundays. Next I tried taking the difference between the workday function value for the start date and the workday value for the end date. I thought that because the workday function accounts for the weekend when adding days, the values it returned would also not include the weekend. For example when April 1st, which is a Friday, is entered into the function, the value that is returned is 40634. I thought that when April 4th, a Monday, was entered it would only be one number higher. Instead it returned 40637. At this point I thought it was impossible and was ready to just give up. However when I looked at the code and really thought through what the workday function was doing, I realized the number I needed wasn't the output from the function but the input.