

Improving Productivity Using Visual Basic for Applications

IV&V Workshop
September 13-15, 2011

Tom Marshall

This presentation consists of L-3 STRATIS general capabilities information that does not contain controlled technical data as defined within the International Traffic in Arms (ITAR) Part 120.10 or Export Administration Regulations (EAR) Part 734.7-11.

Purpose



- To provide an overview of *Visual Basic for Applications* (VBA) so analysts can recognize useful applications to IV&V activities
 - Provide relevant examples of VBA use with IV&V activities
- It is ***not*** intended to be a programming or VBA tutorial
 - Syntax is fairly easy to pick up
 - A Demonstration to be held later will provide a closer look at VBA

What is VBA?



As defined by Wikipedia

Visual Basic for Applications (VBA) is an implementation of Microsoft's event-driven programming language Visual Basic 6 and its associated integrated development environment (IDE), which are built into most Microsoft Office applications, such as Word, Excel, PowerPoint, Access.

- It can be used to control many aspects of an application, including manipulating data contained within the application and working with custom user forms or dialog boxes
- It can be used to control one application from another, e.g. creating a report in Word using data from Excel

But What is VBA Really?



Every part of a MS Office application is made up of 'Objects', and these Objects have 'Properties' that can be manipulated

- In Word, for example, the whole document, individual pages, a paragraph, a sentence, a word, even individual characters are objects
- In Excel, worksheets and cells, among other things, are objects
 - Some object properties are Boolean (TRUE/FALSE). For example, a selection of text can have its BOLD property set to TRUE.
 - In Excel, a range of cells has a 'Merged' property that can be TRUE or FALSE
 - Other properties have 'numerical' or 'text' values

**VBA facilitates interaction with an application's
Objects, Properties and Data**

VB Editor



Project Explorer

Script Editing Window

Properties Window

The screenshot shows the Microsoft Visual Basic Editor interface. On the left is the Project Explorer showing a tree view of the project 'VBAProject (GRAIL L-5 Requirements Analysis)'. Below it is the Properties Window for 'Sheet1 Worksheet', displaying various properties like 'DisplayPageBreaks' and 'EnableAutoFilter'. The main area is the Script Editing Window, which contains VBA code for a 'Sub Combine()' procedure. The code includes logic for finding files, opening workbooks, and updating cell values based on analysis results.

```
Sub Combine()  
Dim ExcelFiles() As String  
On Error Resume Next  
Application.Calculation = xlManual  
CombineSpreadsheets.Show  
CombineSpreadsheets.Show  
If (CombineSpreadsheets.Ok.Tag = "False") Then Exit Sub  
CombineQualityAnalysis = CombineSpreadsheets.CombineQuality  
CombineImplementationAnalysis = CombineSpreadsheets.CombineImplementation  
CombineVerificationAnalysis = CombineSpreadsheets.CombineVerification  
CombineIssueResolutionFlag = CombineSpreadsheets.CombineIssueResolution  
If (Not CombineQualityAnalysis And Not CombineImplementationAnalysis And Not CombineVerificationAnalysis And Not CombineIssueResolutionFlag) Then  
Application.ScreenUpdating = False  
Application.DisplayAlerts = False  
ActiveSheet.ShowAllData  
awb = ActiveWorkbook.Name  
FilePath = "C:\Work\Projects\GRAIL\Requirements\Completed Spreadsheets\  
FilesInPath = Dir(FilePath & "*.xl*")  
If (FilesInPath = "") Then  
MsgBox "No files found"  
Exit Sub  
End If  
Fnum = 0  
Do While (FilesInPath <> "")  
Fnum = Fnum + 1  
ReDim Preserve ExcelFiles(1 To Fnum)  
ExcelFiles(Fnum) = FilesInPath  
FilesInPath = Dir()  
Loop  
For i = 1 To Fnum  
Workbooks.Open FileName:=FilePath & ExcelFiles(i)  
Worksheets("Requirements").Activate  
ActiveSheet.ShowAllData  
perloc = InStr(1, ActiveWorkbook.Name, ".")  
Analyst = Left(ActiveWorkbook.Name, perloc - 1)  
cnt = 2  
Do While (Workbooks(awb).Worksheets("Requirements").Cells(cnt, 2) <> "")  
ReqID = Workbooks(awb).Worksheets("Requirements").Cells(cnt, 2)  
With Columns("B:B")  
Set ans = .Find(ReqID, LookIn:=xlValues, lookat:=xlWhole, MatchCase:=False)  
End With  
If (Not ans Is Nothing) Then  
If (Workbooks(awb).Worksheets("Requirements").Cells(cnt, 11) = Analyst And CombineQualityAnalysis) Then  
For j = 12 To 19  
If (Cells(ans.Row, j).Value <> "") Then Workbooks(awb).Worksheets("Requirements").Cells(cnt, j).Value = Cells(ans.Row, j).Value  
Next j  
If (Workbooks(awb).Worksheets("Requirements").Cells(cnt, 6) = "" And Cells(ans.Row, 6) <> "") Then Workbooks(awb).Worksheets("Requirements").Cells(cnt, 6).Value = Cells(ans.Row, 6).Value  
If (Cells(ans.Row, 21).Value <> "") Then  
Workbooks(awb).Worksheets("Requirements").Cells(cnt, 21).Value = Cells(ans.Row, 21).Value  
Workbooks(awb).Worksheets("Requirements").Cells(cnt, 20).Value = "N"  
End If  
End If  
If (Workbooks(awb).Worksheets("Requirements").Cells(cnt, 24) = Analyst And CombineImplementationAnalysis) Then  
For j = 25 To 29  
If (Cells(ans.Row, j).Value <> "") Then Workbooks(awb).Worksheets("Requirements").Cells(cnt, j).Value = Cells(ans.Row, j).Value  
Next j  
End If  
End If  
End For  
End While  
End For  
End Sub
```

The VB Editor is accessed from the 'Developer' Tab (or Alt-F11)

VBA Application to IV&V



- VBA saves time by automating repetitive IV&V tasks
- VBA improves accuracy by automating tedious tasks
- It is only a tool and is not a substitute for engineering knowledge and judgment applied by an analyst
- When to apply VBA rather than engaging the Software Assurance Tools (SWAT) Group for help?
 - Quick turnaround; need it done right now
 - Straightforward solution
 - Narrow application
 - Something routinely encountered

Example Uses of VBA



- Dealing with software development artifacts
 - Parsing documents into a more useable form, e.g. SRS in PDF
 - Delta artifact analysis, e.g. identifying added, deleted and modified requirements
 - Test script/results “prettifier”
- Correlating data
 - Integrate data from multiple sources, e.g. integrate analysts spreadsheets
 - Merge data, e.g. requirements and verification documentation
 - Identify and display parent/child requirement relationships
- Processing large amounts of data
 - Lint Processor
 - Data post-processing, e.g. telemetry data from tests
 - 1553 Message interpreter
- Extracting and manipulating data from other sources
 - Extract schedule data from MS Project and manipulate using Excel’s rich formulae suite
- Others?

Parting Thoughts



- You don't have to be a highly experienced programmer to effectively use VBA
- One way to learn the keywords, syntax and object properties is to use the macro recorder to record an action
 - The VB code generated by the recorder can be viewed and often customized to suit other needs
- Use the internet
 - Many online tutorials available
 - [microsoft.public.excel.programming](#) in Google Groups
- Build up a code/script “toolbox”

Always be alert for situations where VBA may be useful