Teaching Old Dogs New Tricks- Moving Beyond Excel Estimates

ACEIT Users Workshop
February 1, 2011
Chris Gardiner
Wilson Edwards

Abstract

"Whatever they say about not being able to teach old dogs new tricks, it is patently untrue. Old dogs may not learn as quickly as they did when they were young, but with time and patience, most older dogs can be taught to do anything that a young dog can." ***(petplace.com)***

Some "old dogs" continue to use Excel to build estimates for a variety of reasons. Others casually use ACEIT, use it less efficiently than they could, or revert back to Excel because they believe Excel to be "easier". Regardless of the reasons, "old dogs" can be taught new tricks that will begin to reveal some of the features and capabilities of ACEIT that will make them feel "young" again.

This presentation will address questions as to why some use Excel over ACE; it will show examples of "tricks" that, based on our experience, are often not used because they haven't been discovered or are not understood; it will also show how those "tricks" can be utilized to create more powerful, dynamic estimates.

ALL DATA IN THIS PRESENTATION ARE ENTIRELY NOTIONAL AND DO NOT REPRESENT ACTUAL DATA OF ANY CONTRACTOR OR PROGRAM

Presentation Outline

- Presentation History
- Reasons for Using Excel vs ACEIT
- ACEIT Benefits
- Eliminating Errors
- Using Dates and Date Functions
- Referencing Data Tables in ACE
- Tips and Tricks
- Summary

Presentation History

- **Discussions with Students and other cost estimators revealed extent of Excel use**
- **Surveyed Excel users to identify reasons for Excel preference**
- **Identified common issues and solutions**
- **Obtained Excel Models and Converted to ACEIT**

Reasons for Using Excel vs ACEIT

- **Culture – “It’s the way we have always done it”**
- **Familiar and comfortable with software – “It’s easier”**
- **Unfamiliar with ACEIT; Don’t understand the features, capabilities, and benefits - “Lack of training and experience”**
- **“Effort converting Excel models doesn’t seem worth it”**
- **“ACEIT is more cumbersome for documentation purposes”**
- **“ACEIT too cumbersome for simple estimates”**

Reasons for Using Excel vs ACEIT (cont)

- **“More flexibility to customize the spreadsheet”**
- **“Able to write Macros for files”**
- **“A tool that can be shared among more people”**
- **“Excel’s capabilities are more versatile…very specific templates in Excel”**
- **“Easier to show and explain methodologies to PM”**
- **“I’ve been too busy”**

ACEIT Provides Substantial Benefits

- **Implements Standardized Process**
  - Supports development of consistent, systematic and defendable Life Cycle Cost Estimates
  - Delivers integrated, automated documentation, with complete audit trail
  - Improves estimate review and verification process through consistent model structure
  - Contains industry approved algorithms and databases to model inflation, learning, and phasing
  - Integrates statistical and risk analysis to quantify uncertainty in estimates
  - Enhances quality by eliminating many errors often made in spreadsheets (which frequently go undetected)
- **Provides Flexibility to Model any System Type**
  - Unlimited flexibility to model any type of system linking all life cycle phases, and facilitate any type of Analysis of Alternatives.
  - Automated and customizable reports
- **Integrates with Other Applications Through an Open Platform**
  - Ability to link to virtually any other tool
  - Robust Application Programming Interface (API) to facilitate electronic interaction
- **Reduces Management Challenges**
  - Structured modeling platform shortens time for ACE users to learn a new model
  - Eases organization-wide distribution of key standards (WBS, inflation, etc)

Eliminating Errors

Eliminating Errors

- **Excel Cost Estimate Model Example**
- **Reprogramming the Thinking Process**
- **Conversion to ACEIT**
- **ACE's Error Log**

Excel Cost Estimate Model Example

Excel Model Items of Note

- Concatenate used in Excel model for cell reference name (unique ID) requires multiple steps
- Excel requires formulas on parent rows
- Excel model inflation requires separate inflation worksheets/tables and multiple look up equations vs BY and units
- Documentation
- All Excel phasing requires separate phasing tables and multiple look up equations
- Excel dollar units fixed without separate tables and calculations

Errors in Excel Estimate - Common

- **Many errors often go undetected in Excel**
- **Examples of errors in example Excel Model**
  - Production Kit costs multiplied by incorrect Qty row
    - Unit 2 Kit multiplied by Unit 3 Qty
    - Unit 3 Kit Multiplied by Unit 2 Qty
  - Install costs for Unit 2 and Unit 3 were omitted
  - Maintainer Training referenced the wrong schedule cell and costs were calculated one year early
  - Allocations of total costs for travel and ECO's did not equal 100%

Excel Cost Estimate Model Converted to ACEIT

Conversion of Excel to ACEIT
Items of Note

- **WBS/CES indenture and numbering**
- **ACE unique ID reference vs annual cell reference**
- **Cost Interpretation (BY; Units) and Approp replace Excel tables to convert units, inflate, and convert BY to TY**
- **Phasing methodologies replace Excel tables and formulas in every cell**
- **Documentation - convenience and usefulness**

Conversion of Excel to ACEIT Items of Note (cont)

- **Use of functions to automate model**
  - ACE Dates and Date Functions allow for schedule sensitivity and linking
  - FYYR Functions
- **Integrated risk analysis**
- **Reports easily created and modified without creating new tables and redundant formulas**
- **ACE's error log identifies the problem, or potential problem, and where it is located**

Conversion of Excel to ACEIT Items of Note (cont)

- **It takes only one-fourth to one-tenth the time to perform the same analysis in ACE that is required for the same analysis in Excel**

ACE's Error Log

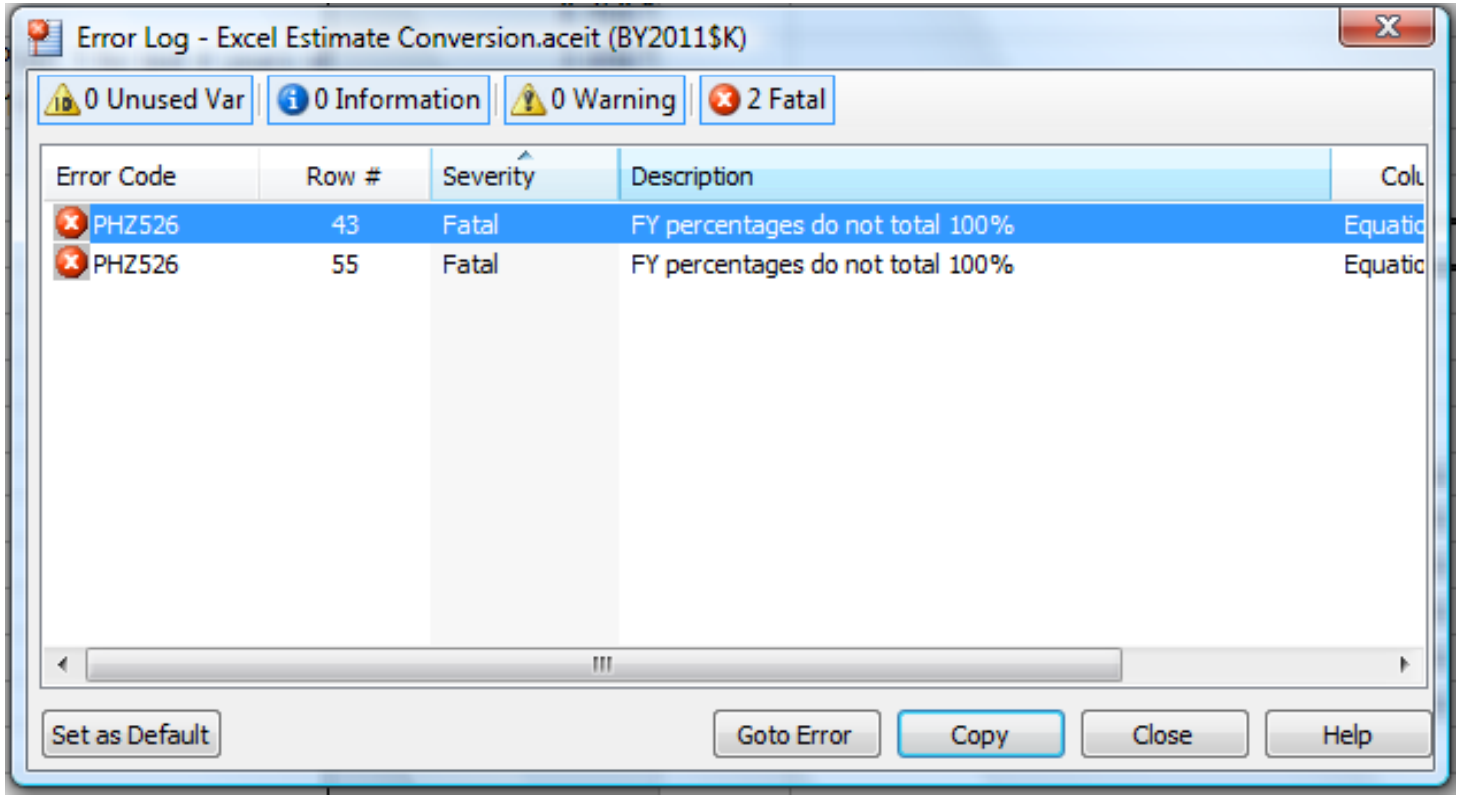
ACE's Error Log

- **ACEIT Enhances quality by eliminating many errors often made in spreadsheets (which frequently go undetected)**
- **Common ACE Error Messages**

ERROR CODE	SEVERITY	DESCRIPTION	TYPICAL CAUSE
MTH550	Warning	WBS has unspecified methodology	There is no equation or throughputs entered for the row.
MTH562	Warning	Unused variable "ID"	The Unique ID is not being used in any equations.
MTH558	Fatal	Variable "ID" defined more than once	There is more than one row with the same Unique ID.
PHZ628	Fatal	Item unphased but summed with phased items	There is not a phasing method on the row.
PHZ567	Fatal	"R" Method w/o learning specified	There is an R method on a row with no learning inputs.
PHZ526	Fatal	FY percentages do not total 100%	Allocations entered in the Fiscal Years do not add to 100.
ADJ678	Fatal	Cost without an appropriation	Verify that the row is a cost, if it is a cost, the appropriation is missing.

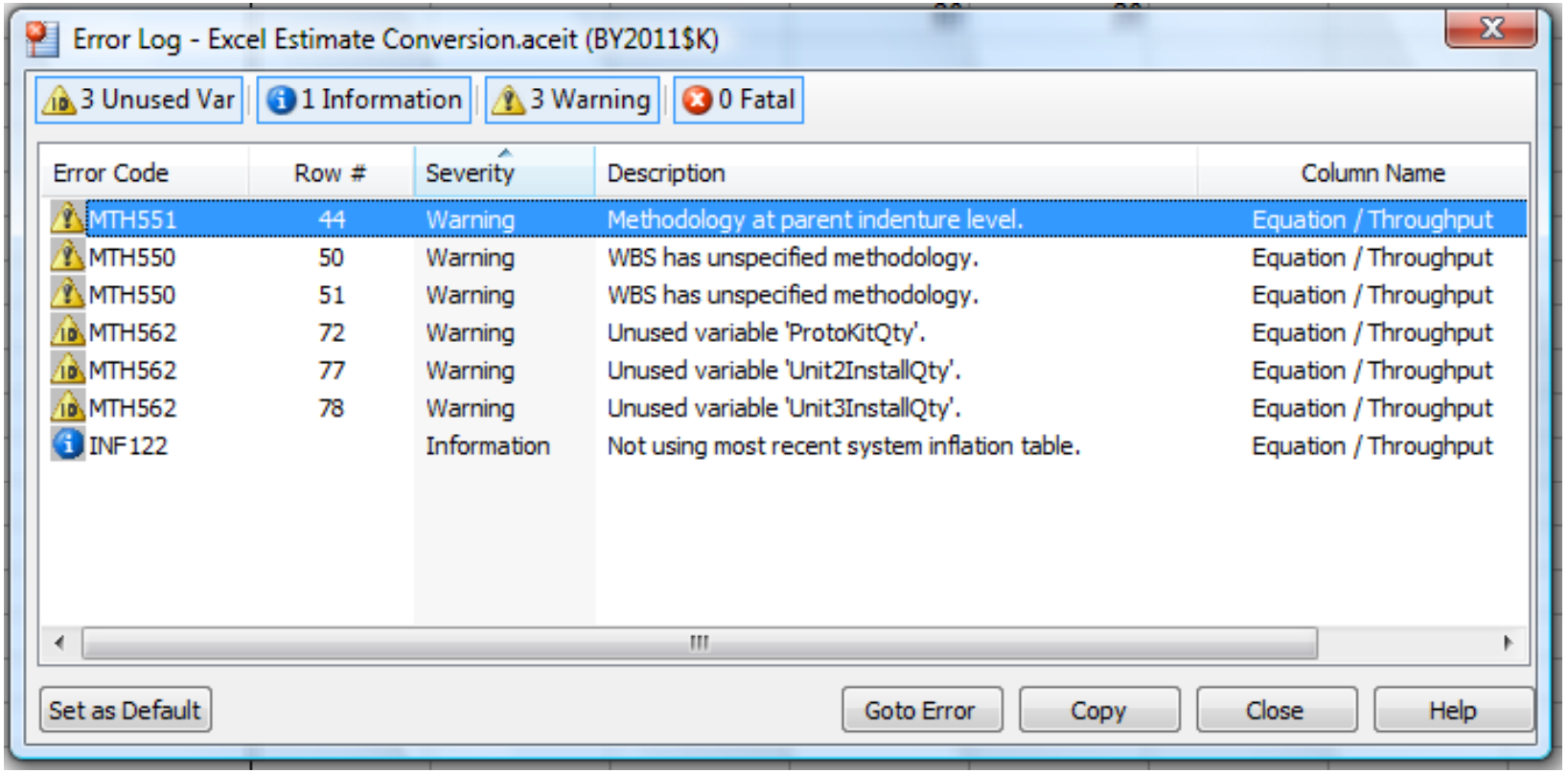
Conversion of Excel Model to ACE – Error Log

- Duplicating the errors found in the Excel Model in our ACE Sessions resulted in the following Fatal Errors



Conversion of Excel Model to ACE – Error Log (cont)

- Duplicating the errors found in the Excel Model in our ACE Sessions resulted in the following error messages



Using Dates and Date Functions

Milestone Date Sections - DateAdd Function

- **Example with Duration in Months:**

	WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method	Equation / Throughput
64	*INPUT VARIABLES		*IN_VAR			
65	*Milestone Dates					
66	Development Start Date		DevStart	01OCT2010 *	C	01OCT2010
67	Development Duration in Months		DevDur	24.000 *	C	24
68	Development Finish Date		DevFinish	30SEP2012 *	C	DateAdd(DevStart, 0, DevDur, -1)
69	Production Start Date		ProdStart	01OCT2011 *	C	DateAdd(DevStart, 1)
70	Production Finish Date		ProdFinish	30SEP2017 *	C	DateAdd(ProdStart, 6, 0, -1)

- **Use the DateAdd function to calculate the finish date**

Calculating Start and Finish Dates from Other Element Schedules

- **Time Phased Buy Quantity**

	WBS/CES Description	Phasing Method	FY 2011	FY 2012	FY 2013	FY 2014	FY 2015	FY 2016	FY 2017
72	*Production Schedules								
73	Prototype Kits	IS		1					
74	Unit 1, Production Kit	IS			5	5	5	5	
75	Unit 2, Production Kit	IS			10	10	10	10	
76	Unit 3, Production Kit	IS			7	7	7	4	
77	Unit 1 Install	IS			3	5	5	5	2

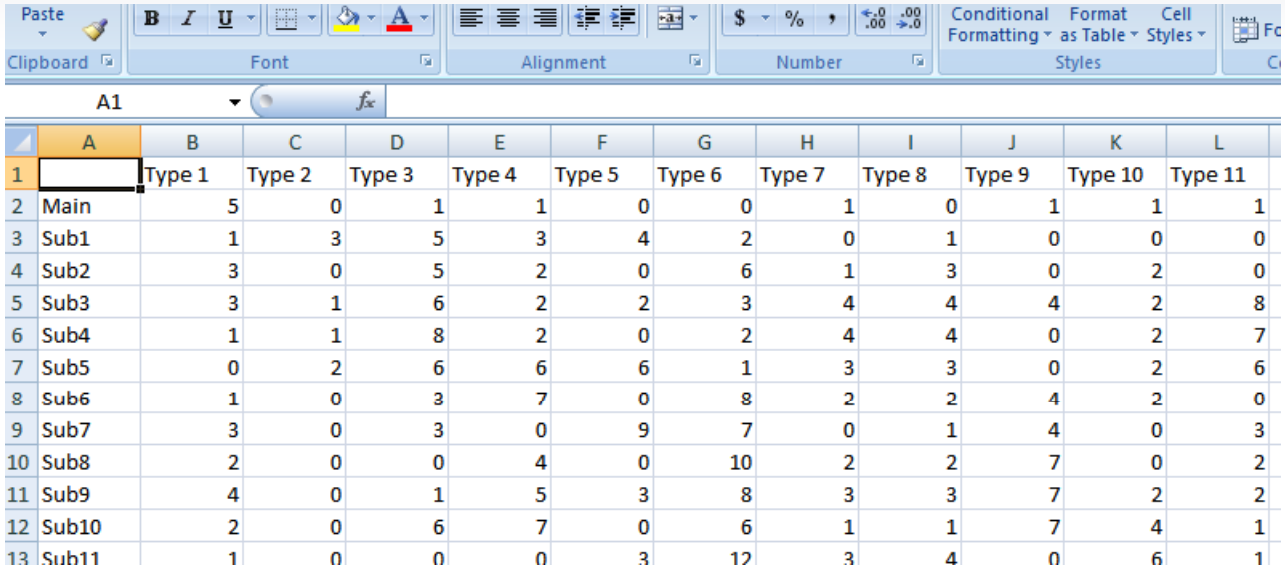
- **Use DateOf function to calculate start and end dates**

	WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method	Equation / Throughput
65	*Milestone Dates					
69	Production Start Date		ProdStart	01OCT2011 *	C	DateOf(FYCFirstYr(@ProtoKitQty))
70	Production Finish Date		ProdFinish	30SEP2017 *	C	DateOf(FYCLastYr(@Unit1InstallQty)+1)-1

- **DateOf Syntax**
  - DateOf (Year, [Month], [Day], [Year_Type])
  - When used with only the year parameter, this function returns the first day of the fiscal year

Referencing Data Tables in ACE

Tables in ACE



	A	B	C	D	E	F	G	H	I	J	K	L
1		Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9	Type 10	Type 11
2	Main	5	0	1	1	0	0	1	0	1	1	1
3	Sub1	1	3	5	3	4	2	0	1	0	0	0
4	Sub2	3	0	5	2	0	6	1	3	0	2	0
5	Sub3	3	1	6	2	2	3	4	4	4	2	8
6	Sub4	1	1	8	2	0	2	4	4	0	2	7
7	Sub5	0	2	6	6	6	1	3	3	0	2	6
8	Sub6	1	0	3	7	0	8	2	2	4	2	0
9	Sub7	3	0	3	0	9	7	0	1	4	0	3
10	Sub8	2	0	0	4	0	10	2	2	7	0	2
11	Sub9	4	0	1	5	3	8	3	3	7	2	2
12	Sub10	2	0	6	7	0	6	1	1	7	4	1
13	Sub11	1	0	0	0	3	12	3	4	0	6	1

- **Look familiar??? We are given data in this format time and time again**
- **Some users think it is relatively simple to continue using and manipulating the data in the form it was given**
- **But if…..**

Tables in ACE (cont)

- **The Yearly phasing workscreen has more uses than many have previously been exposed to**
- **You can store a table of data (parts list, technical data, site component tables, etc.) that has nothing to do with the fiscal years**
- **When working with fiscal year data you should use FYCVal() function, but when the data is entered as a table, the Matrix functions are easier ways to utilize the data**

Utilizing data tables in ACE

- **Accessing the data is similar in function**
  - Excel uses these functions:
    - Direct reference (“=C4”, etc.)
    - Lookup (Lookup, Vlookup, Hlookup)
    - SumIf, SumIfs
    - Combinations of above
  - ACE uses these functions to retrieve data from particular fiscal year columns:
    - FYCVal
    - Matrix (MatVal, MatColCol, MatColTot, etc.)
    - Vlookup
    - Dot Notation (X.1999, X.FYTot)

Why use Matrices in ACE?

Benefits of using Matrices

- **Organization - Easier to logically arrange data**
- **Input Flexibility – Data inputs are easily modified**
  - Changing configurations
  - Frequently updated buy quantities/schedule
- **Scalability - Easy to increase/decrease the size of matrix**
- **Repeatability and ease of model building - Allow you to use the same equation on multiple rows**

ACE Matrices

- **Setting up matrices**
  - Use a comment row (*) to describe the matrix and establish column ‘table’ headings –technically optional-
  - The “Parent” row will have a Unique ID to reference the matrix
  - The “Child” rows are the ‘table’ rows and are indented
  - Establish phasing, as applicable

WBS/CES Description	Unique ID	Phasing Method	FY 2006	FY 2007	FY 2008	FY 2009
*Site2 Matrix of Parts			Type 1	Type 2	Type 3	Type 4
Site2	Site2PartsQty					
Main	Site2MainQty	I	5	0	1	1
Sub1	Site2AQty	I	1	3	1	3
Sub2	Site2BQty	I	3	0	5	2
Sub3	Site2CQty	I	3	1	6	2
Sub4	Site2DQty	I	1	1	8	2

FYCVal Function

- **FYCVal retrieves a value from a particular year on the yearly phasing workscreen**
  - Inputs:

WBS/CES Description	Unique ID	FY 2003	FY 2004	FY 2005
****** Cost by Pay Category by Personnel Type**		Pay	Training	Other
Yearly Rate Matrix	LRate			
Officer		55	1.2	10
Enlisted		35	0.9	5
Civilian		45	2.5	0
Other		12	.1	0

  - Output: FYCVal(@LRate+3, FYFirst)

(@Reference, Year)

Cost Element	Total
Civilian Pay Rate	45.0

FYIVal also retrieves data from the yearly phasing workscreen

MatVal Function

- **MatVal retrieves a value from a matrix**
  - Inputs:

WBS/CES Description	Unique ID	FY 2003	FY 2004	FY 2005
****** Cost by Pay Category by Personnel Type**		Pay	Training	Other
Yearly Rate Matrix	LRate			
Officer		55	1.2	10
Enlisted		35	0.9	5
Civilian		45	2.5	0
Other		12	.1	0

  - Output: MatVal(@LRate, 3, 1)

(@Matrix, Row, Column)

Cost Element	Total
Civilian Pay Rate	45.0

Matrix Functions

- **Instead of inserting extra rows for intermediate calculations, MatColRow() will give the sum of products:**

WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method	Equation / Throughput	Fiscal Year	Units	Start Date
Site1		Site1Item1$	$ 19,000.513 *					
Main	3020	Site1MainItem1$	$ 278.735 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 1))			
Element1			$ 8,940.323 *					
Sub1	3020	Site1Sub1Item1$	$ 306.010 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 2))			
Sub2	3020	Site1Sub2Item1$	$ 477.744 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 3))			
Sub3	3020	Site1Sub3Item1$	$ 535.683 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 4))			
Sub4	3020	Site1Sub4Item1$	$ 476.839 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 5))			
Sub5	3020	Site1Sub5Item1$	$ 561.147 *	F	(MATCOLROW(10, @Item1Costs$, @Site1PartsQty, 6))			

WBS/CES Description	Unique ID	FY 2006	FY 2007	FY 2008	FY 2009	FY 2010	FY 2011	FY 2012	FY 2013	FY 2014	FY 2015	FY 2016
*Site1 Matrix of Parts		Type 1	Type 2	Type 3	Type 4	Type 5	Type 6	Type 7	Type 8	Type 9	Type 10	Type 11
Site1	Site1PartsQty											
Main		5	0	1	1	0	0	1	0	1	1	1
Sub1		1	3	5	3	4	2	0	1	0	0	0
Sub2		3	0	5	2	0	6	1	3	0	2	0
Sub3		3	1	6	2	2	3	4	4	4	2	8
Sub4		1	1	8	2	0	2	4	4	0	2	7
Sub5		0	2	6	6	6	1	3	3	0	2	6
Sub6		1	0	3	7	0	8	2	2	4	2	0
Sub7		3	0	3	0	9	7	0	1	4	0	3
Sub8		2	0	0	4	0	10	2	2	7	0	2
Sub9		4	0	1	5	3	8	3	3	7	2	2

WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method	Equation / Throughput
Item1 Costs (vector for Matrix function)		Item1Costs$	$ 496.489 *		
Type 1	3020		$ 26.841 *	C	Type1$
Type 2	3020		$ 22.769 *	C	Type2$
Type 3	3020		$ 18.673 *	C	Type3$
Type 4	3020		$ 14.705 *	C	Type4$
Type 5	3020		$ 13.447 *	C	Type5$
Type 6	3020		$ 9.375 *	C	Type6$
Type 7	3020		$ 3.545 *	C	Type8$
Type 8	3020		$ 0.853 *	C	Type7$
Type 9	3020		$ 1.560 *	C	Type9$
Type 10	3020		$ 106.050 *	C	Type10$
Type 11	3020		$ 278.673 *	C	Type11$

MatTotTot Function

- **MatTotTot multiplies two vectors together**
  - Example: What is the Total Personnel Cost per Year (across all four pay groups)?
  - Inputs:

WBS/ CES Description	Unique ID	Total
**** Pay Rate by Personnel Type		
Yearly Rate Matrix	PRate	147.0
Officer		55.0
Enlisted		35.0
Civilian		45.0
Other		12.0
*** Personnel Requirement Inputs		
Steady State Personnel Requirement	SS_P	340.0
Officer		28.0
Enlisted		150.0
Civilian		150.0
Other		12.0

  - Output: MatTotTot(4, @SS_P, @PRate)

Cost Element	Unique ID	Total
Total Cost of Personnel per Year		13,684.0

Total = (55*28) + (35*150) + (45*150) + (12*12) = $13,684

MatColTot Function (cont)

- **MatColTot multiplies result from a selected column of a matrix by a vector**
  - Example: What is the Total Personnel Cost (across all four pay groups)? – Pay data is in a FY column not the Total
  - Inputs:

WBS/ CES Description	Unique ID	FY 1	FY 2	FY 3
**** Cost by Pay Category by Personnel Typ		Pay	Training	Other
Yearly Rate Matrix	LRate			
Officer		55	1.2	10
Enlisted		35	0.9	5
Civilian		45	2.5	0
Other		12	.1	0

WBS/ CES Description	Unique ID	Total
*** Personnel Requirement Inputs		
Steady State Personnel Requirement	SS_P	340.0
Officer		28.0
Enlisted		150.0
Civilian		150.0
Other		12.0

  - Output: MatColTot(4, @SS_P, @LRate, 1)

Cost Element	Total
Pay	$ 13,684.0

Total = (55*28) + (35*150) + (45*150) + (12*12) = $13,684

MatColCol Function

- **MatColCol multiplies a year of an FY-dependent matrix by a single column in another matrix**
  - Example: What is the Total Personnel Cost per year?
  - Inputs:

Cost Element	Unique ID	FY 2003	FY 2004	FY 2005	FY 2006	FY 2007
*** Personnel Requirements						
Personnel Requirements by Year	People	135.0	340.0	340.0	340.0	340.0
Officer		11.0	28.0	28.0	28.0	28.0
Enlisted		60.0	150.0	150.0	150.0	150.0
Civilian		60.0	150.0	150.0	150.0	150.0
Other		4.0	12.0	12.0	12.0	12.0

**** Cost by Pay Category by Personnel Typ		Pay	Training	Other
Yearly Rate Matrix	LRate			
Officer		55	1.2	10
Enlisted		35	0.9	5
Civilian		45	2.5	0
Other		12	.1	0

  - Output: MatColCol(4, @People, @LRate, 1)

Cost Element	Unique ID	FY 2003	FY 2004	FY 2005	FY 2006	FY 2007
Pay		$ 5,453.0	$ 13,684.0	$ 13,684.0	$ 13,684.0	$ 13,684.0

FY2003 = (11*55) + (60*35) + (60*45) + (4*12) = $5,453

MatColRow Function

- **MatColRow multiplies a year of an FY-dependent matrix by a single row in another matrix**
  - Example: Personnel Rate Matrix is now transposed - What is the Total Personnel Cost per year?
  - Inputs:

Cost Element	Unique ID	FY 2003	FY 2004	FY 2005	FY 2006	FY 2007
*** Personnel Requirements						
Personnel Requirements by Year	People	135.0	340.0	340.0	340.0	340.0
Officer		11.0	28.0	28.0	28.0	28.0
Enlisted		60.0	150.0	150.0	150.0	150.0
Civilian		60.0	150.0	150.0	150.0	150.0
Other		4.0	12.0	12.0	12.0	12.0

Table Transposed

**** Cost by Personnel Type by Pay Category		Officer	Enlisted	Civilian	Other
Yearly Rate Matrix	LRate2				
Pay		55	35	45	12
Training		1.2	0.9	2.5	.1
Other		10	5	0	0

  - Output: MatColRow(4, @People, @LRate2, 1)

Cost Element	Unique ID	FY 2003	FY 2004	FY 2005	FY 2006	FY 2007
Pay		$ 5,453.0	$ 13,684.0	$ 13,684.0	$ 13,684.0	$ 13,684.0

FY2003 = (11*55) + (60*35) + (60*45) + (4*12) = $5,453

Useful Matrix Tips

- **Start small, then build on it to verify functionality**
- **The System By Site Wizard can help!**
- **Dynamic Equation Columns (DECs) can be used to automate and add further functionality to your matrices**
- **Color code matrix-related data to make it easier to find**
  - Text Color
  - Highlighting

The VLookup & StepVal Functions

The VLookup Function

- The Vertical Lookup Function searches for a value stored in the first FY column, once found, returns the value on the located row from the specified column
- VLookup and StepVal perform similar operations, the difference being that StepVal is limited to returning values only from one specified vector (@Y), and is oriented horizontally
- VLookup allows for a matrix to be specified and values returned from any vector in the matrix

VLookup Example

- **VLookup Function Syntax:**

VLookup(LookupVal, @DataTable, ColIndex, NumRows)

Example:

WBS/CES Description	Unique ID	Point Estimate	Equation / Throughput	FY 2003	FY 2004	FY 2005
*My Program Estimate	*Estimate					
***Syntax**			**VLookup(lookup_value, @DataTable, col_index, num_rows)**			
Number of Fuel Tanks needed for HW with Qty = 8	Row	2.000 *	VLookup(8, @HWReq, 2, num_rows)			
Number of Fuel Tanks needed for HW with Qty = 10		2.000 *	VLookup(10, @HWReq, 2, num_rows)			
ManPower needed for HW with Qty 17		8.000 *	VLookup(17, @HWReq, 3, num_rows)			
ManPower needed for HW with Qty 50		8.000 *	VLookup(50, @HWReq, 3, num_rows)			
*INPUT VARIABLES	*IN_VAR					
*** **Fuel Tanks and Manpower needed by Qty**				**Qty**	**Fuel Tanks**	**Manpower**
HW Fuel and Manpower Matrix	HWReq	0.000 *				
HW Qty 1-5			[Input Throughput]	5	1	2
HW Qty 6-10			[Input Throughput]	10	2	4
HW Qty 11-15			[Input Throughput]	15	4	6
HW Qty 16-20			[Input Throughput]	20	6	8
Number of Rows	num_rows	4.000 *	4			

The StepVal Function

- **Using previous scenario for determining Fuel Tanks and Manpower based on HW Qty**

*** Fuel Tanks and Manpower needed by Qty				Qty	Fuel Tanks	Manpower
HW Fuel and Manpower Matrix	HWReq	0.000 *				
HW Qty 1-5			[Input Throughput]	5	1	2
HW Qty 6-10			[Input Throughput]	10	2	4
HW Qty 11-15			[Input Throughput]	15	4	6
HW Qty 16-20			[Input Throughput]	20	6	8
Number of Rows	num_rows	4.000 *	4			

- **The StepVal function allows you to simplify and automate, especially for multiple year inputs, or case analyses**

WBS/CES Description	Unique ID	Point Estimate	Equation / Throughput	Phasing Method	FY 2010	FY 2011	FY 2012	FY 2013
Price point breaks	PriceStep		[Input Throughput]	I	5	10	15	20
Fuel Tanks	FuelTankQty		[Input Throughput]	I	1	2	4	6
Manpower	ManpowerQty		[Input Throughput]	I	2	4	6	8

The StepVal Function (cont)

- **StepVal Function Syntax:**

  StepVal(xval, @x, @f_of_x, num_steps)

- **Automate relevant information**
- **Establish criteria and desired functionality**

WBS/CES Description	Unique ID	Point Estimate	Equation / Throughput	Phasing Method	FY 2010	FY 2011	FY 2012
Hardware Qty	HWQty		[Input Throughput]	IS	5	6	10
Fuel Tank Qty (Based on Hardware Qty)			StepVal(HWQty, @PriceStep, @FuelTankQty,BreakQty)	F			
Manpower Qty (Based on Hardware Qty)			StepVal(HWQty, @PriceStep, @ManpowerQty,BreakQty)	F			
Price point breaks	PriceStep		[Input Throughput]	I	5	10	15
Number of price point breaks	BreakQty		FYCLastYr(@PriceStep) - FYCFirstYr(@PriceStep) + 1	C			
Fuel Tanks	FuelTankQty		[Input Throughput]	I	1	2	4
Manpower	ManpowerQty		[Input Throughput]	I	2	4	6

Tips and Tricks

Tips & Tricks

- **Appropriation Choices**
- **ACEIT eNews Tips of the Month**
  - OpCycle Function vs FYRepeat Function
- **Definition Cleanup**
- **ACEIT.com**
- **Convergence**
- **Templates**

Tips & Tricks – Appropriation Choices

Appropriation Choices

- **In the published system (US Government) inflation tables the same appropriation number (i.e. 3400) is used for multiple indices**
  - 3400 - GS & Wage Board Pay
  - 3400 - O&M Non-Pay, Non-POL
- **ACE distinguishes which index to use based on appropriation choices**
  - 3400 - O&M Non-Pay, Non-POL
  - 3401 - GS & Wage Board Pay
- **Be aware when using codes in ACE**
  - Using 3400 for GS & Wage Board Pay would use the wrong index

Appropriation Choices (cont)

USAF Raw Inflation Indices Based on OSD Raw Inflation Rates Base Year (FY) 2011							
					General	Operations	Research,
		Military Compensation			Services	& Maint.	Develop.,
	Pay	Other			& Wage	Non-Pay,	Testing,
Fiscal	Base	Expenses	Total	Retirement	Board Pay	Non-POL	Evaluation
Year	(3500)	(3500)	(3500)	(3500)	(3400)	(3400)	(3600)
2005	0.837	0.863	0.840	0.713	0.848	0.891	0.891
2006	0.864	0.886	0.866	0.709	0.876	0.918	0.918
2007	0.885	0.903	0.887	0.726	0.897	0.943	0.943
2008	0.913	0.930	0.915	0.820	0.925	0.966	0.966
2009	0.948	0.960	0.949	0.863	0.960	0.980	0.980
2010	0.981	0.988	0.982	0.981	0.985	0.989	0.989
2011	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2012	1.021	1.020	1.021	1.021	1.021	1.016	1.016

Different Indices – Same Appropriation

Appropriation Choices (cont)

WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method
***Wage Rates**				
Annual GS-7 Step 5	3401	GS7Ann$	$ 44.651 *	C
Annual GS-9 Step 5				
Annual GS-11 Step 5				
Annual GS-12 Step 5				
Annual GS-14 Step 5				
Annual Program Manager				
Annual Engineer				
Annual Equipment Specialist				
Annual Technical Writer				
Annual Provisioner				
Annual Manager	3401	MgrAnn$	$ 153.920 *	C

3600 - AIR FORCE - RSCH, DEV, TEST & EVAL
3080 - AIR FORCE - OTHER PROCUREMENT
3400 - AIR FORCE - O&M - NON-PAY, NON-POL
3401 - AIR FORCE - O&M - GS & WB PAY ONLY
3010 - AIR FORCE - AIRCRAFT PROCUREMENT
3020 - AIR FORCE - MISSILE PROCUREMENT
3300 - AIR FORCE - MILITARY CONSTRUCTION
3402 - AIR FORCE - O&M - FUEL
3500 - AIR FORCE - MILITARY PERSONNEL - TOTAL
3501 - AIR FORCE - MILITARY PERSONNEL - PAY BASE
3502 - AIR FORCE - MILITARY PERSONNEL - OTHER EXPENSES
3503 - AIR FORCE - MILITARY PERSONNEL - RETIREMENT
3730 - AIR FORCE - MILCON - AF RESERVE

ACE Distinguishes which index to use based on appropriation choices

Tips & Tricks – ACEIT eNews Tips of the Month

OpCycle vs FYRepeat Functions

Function	OpCycle	FYRepeat
Description	Calculates a cyclical schedule where a value is repeated every x years for a specified number of cycles	Repeats a schedule every x years a specified number of times
Parameters	Value – Single value or C-phased variable	@var – Row where time-phased schedule is input or calculated
	StartYear – First Year to repeat value	NumTimes – Number of times to repeat the schedule
	CycleYears – Number of years between value	RepeatSize – Number of years before beginning to repeat the schedule again
	Multiplier – (optional) – Increase the value by some percentage each cycle	FY – (optional) – Retrieves a specific year of the schedule
	MaxCycles – (optional) – Repeat the value a specific number of times	
Phasing	F method	F method

OPCycle Function

Inputs:

WBS/CES Description	Approp	Unique ID	Point Estimate	Phasing Method	Equation / Throughput	Fiscal Year	Units	Start Date	Finish Date
*OPCycle Estimate		*Estimate							
Schedule 1			500.000 *	F	OpCycle(Val,2012,2,1)				
Schedule 2			331.000 *	F	OpCycle(Val,2012,4,1.1)				
Schedule 3			400.000 *	F	OpCycle(Val,2012,2,1,4)				
Schedule 4			300.000 *	F	OpCycle(Val,2012,2,1)				2017
*INPUT VARIABLES		*IN_VAR							
Single Value		Val	100.000 *	C	100				

Results:

WBS/CES Description	Total	FY 2011	FY 2012	FY 2013	FY 2014	FY 2015	FY 2016	FY 2017	FY 2018	FY 2019	FY 2020
*OPCycle Estimate											
Schedule 1	500.000		100.000		100.000		100.000		100.000		100.000
Schedule 2	331.000		100.000				110.000				121.000
Schedule 3	400.000		100.000		100.000		100.000		100.000		
Schedule 4	300.000		100.000		100.000		100.000				

FYRepeat Function

Inputs:

*INPUT VARIABLES		*IN_VAR			
Buy Schedule		Sch	6.000 *	IS	[Input Throughput]
Repeat 2 times, every 4 years			12.000 *	F	FYREPEAT(@Sch, 2, 4)
Repeat 2 times, every 4 years with a 2 year lead			12.000 *	F	FYREPEAT(@Sch, 2, 4, FYYR+2)
Repeat 3 times, every 2 years			18.000 *	F	FYREPEAT(@Sch, 3, 2)
Repeat 3 times, every 3.25 years			18.000 *	F	FYREPEAT(@Sch, 3, 3.25)

Results:

WBS/CES Description	Total	FY 2011	FY 2012	FY 2013	FY 2014	FY 2015	FY 2016	FY 2017	FY 2018	FY 2019	FY 2020	FY 2021	FY 2022
Buy Schedule	6.000			1.000	3.000	2.000							
Repeat 2 times, every 4 years	12.000			1.000	3.000	2.000		1.000	3.000	2.000			
Repeat 2 times, every 4 years with a 2 year lead	12.000	1.000	3.000	2.000		1.000	3.000	2.000					
Repeat 3 times, every 2 years	18.000			1.000	3.000	3.000	3.000	3.000	3.000	2.000			
Repeat 3 times, every 3.25 years	18.000			1.000	3.000	2.000	0.750	2.500	2.250	1.000	2.000	2.500	1.000

OpCycle vs FYRepeat Functions

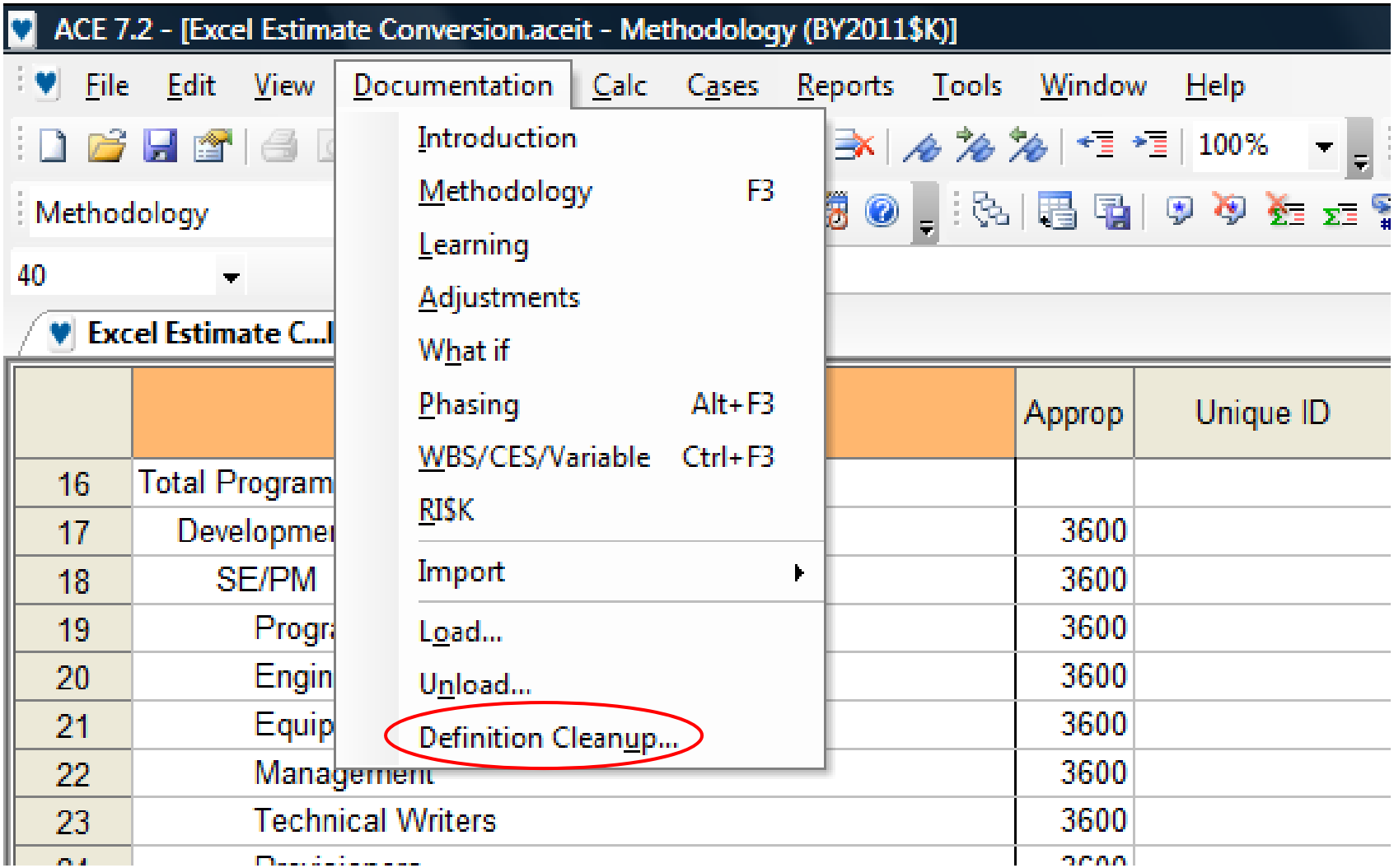
- **OpCycle Function**
  - Use if you want to repeat a single value
- **FYRepeatFunction**
  - Use if you want to repeat a stream of values

Tips & Tricks - Definition Cleanup

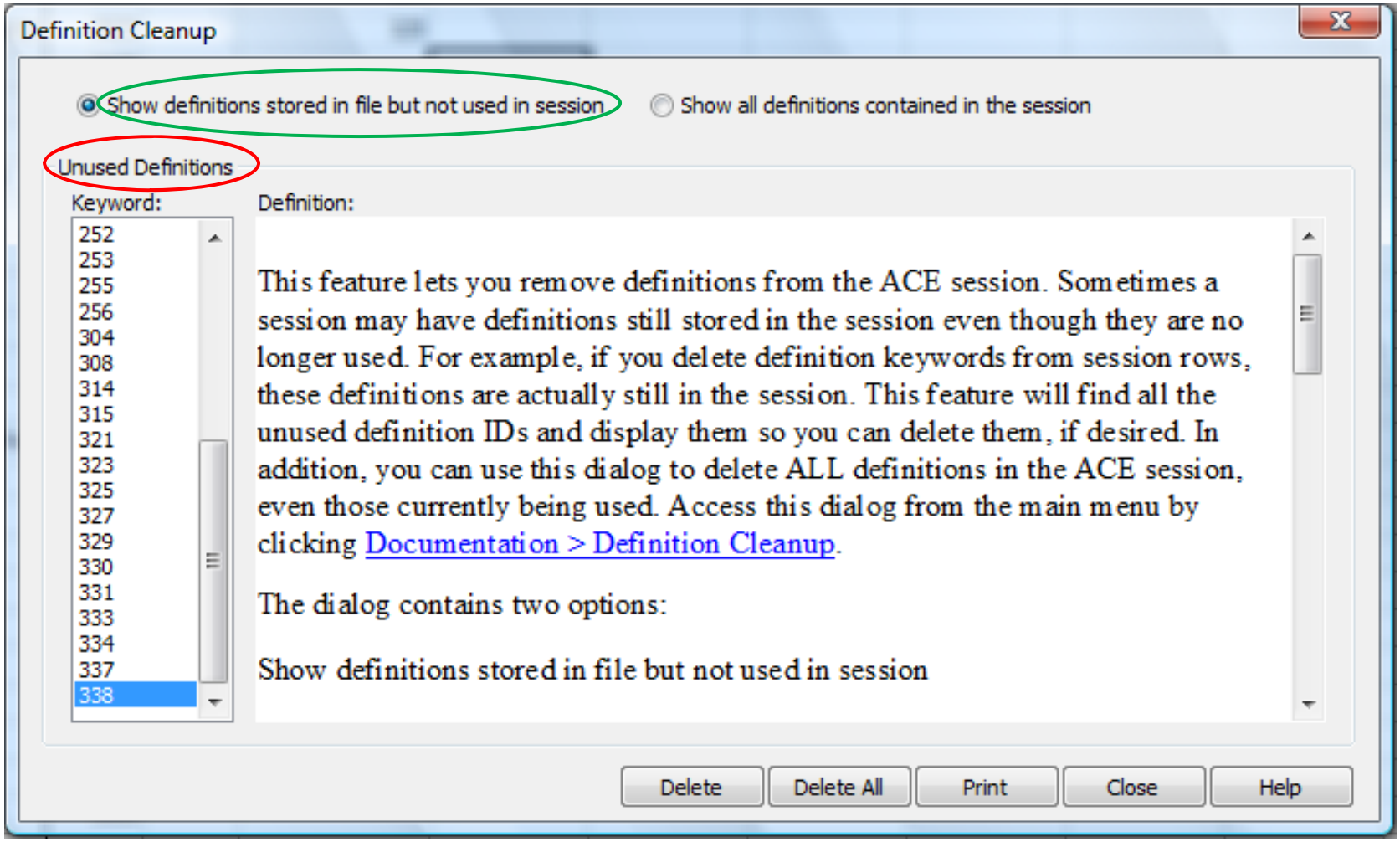
Definition Cleanup

- If you delete definition keywords from session rows, these definitions are actually still in the session
- If significant in size, these unused definitions can require significantly longer calculation times
- This feature lets you remove definitions from the ACE session

Definition Cleanup (cont)



Definition Cleanup (cont)

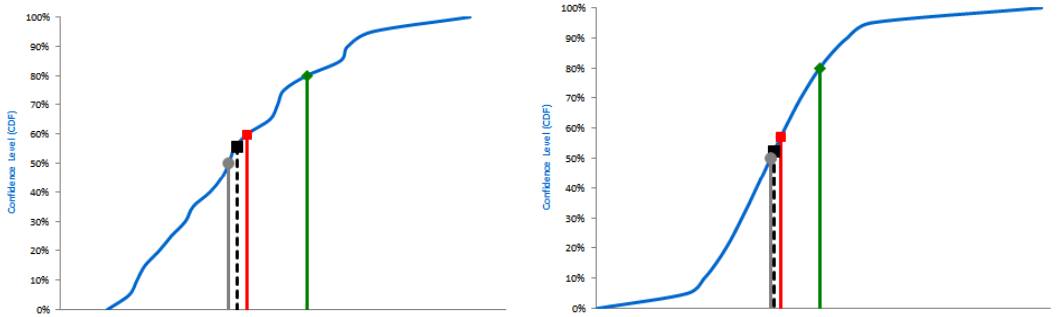


Tips & Tricks - ACEIT.com

Tips & Tricks- Convergence

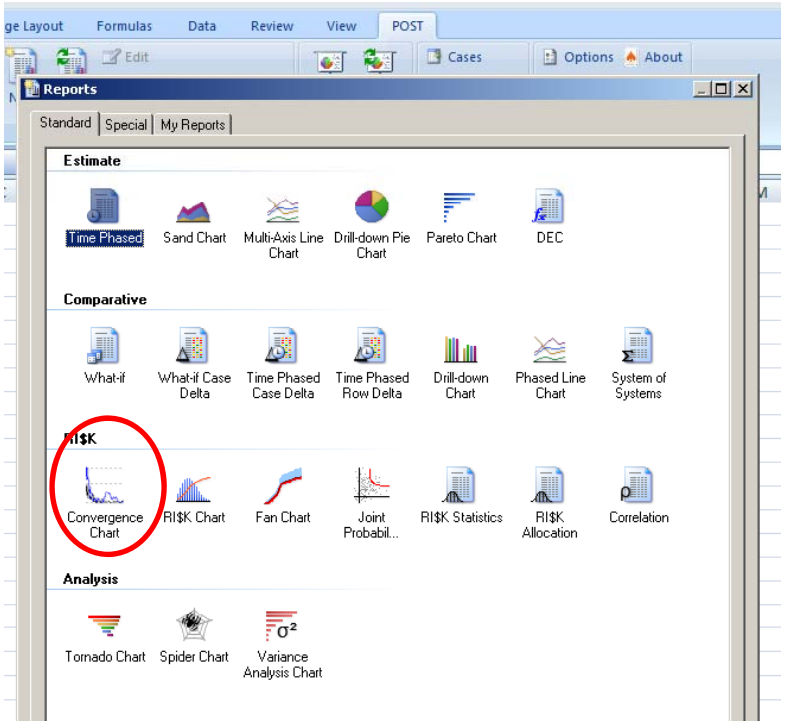
Convergence

- **How many iterations is enough?**
  - From discussions with other estimators, “enough” is
    - “making the S-Curve smooth”
    - 10,000
    - 100,000 (?!!??!)
    - “I don’t know”
- **Is there an *easy* way to find out how many iterations is appropriate for my specific cost model?**



Convergence (cont)

- **The Convergence Chart in POST**
  - Provides guidance on how many iterations should be used in the uncertainty analysis
  - "A good rule of thumb is that once your desired confidence level result stabilizes to within 0.5 percent, the number of iterations corresponding to this is how many you should run." -POST Help

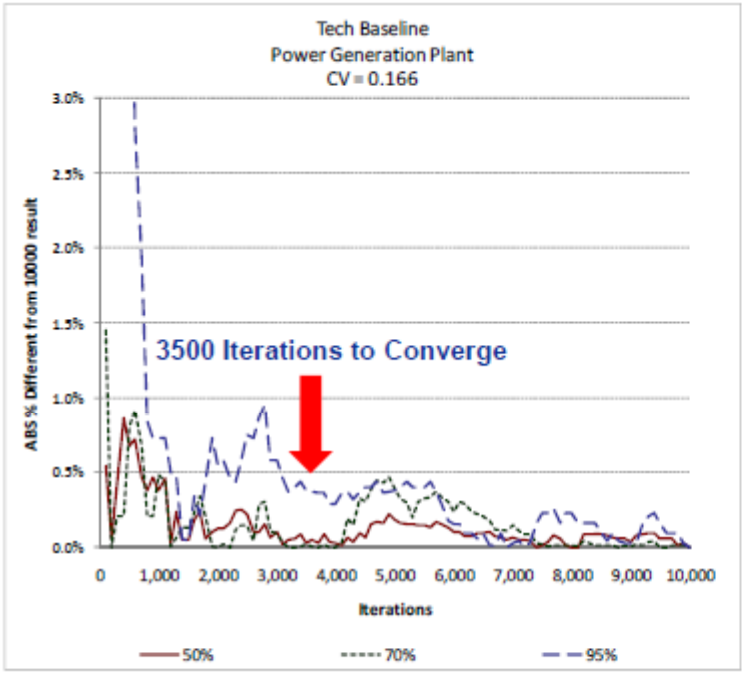


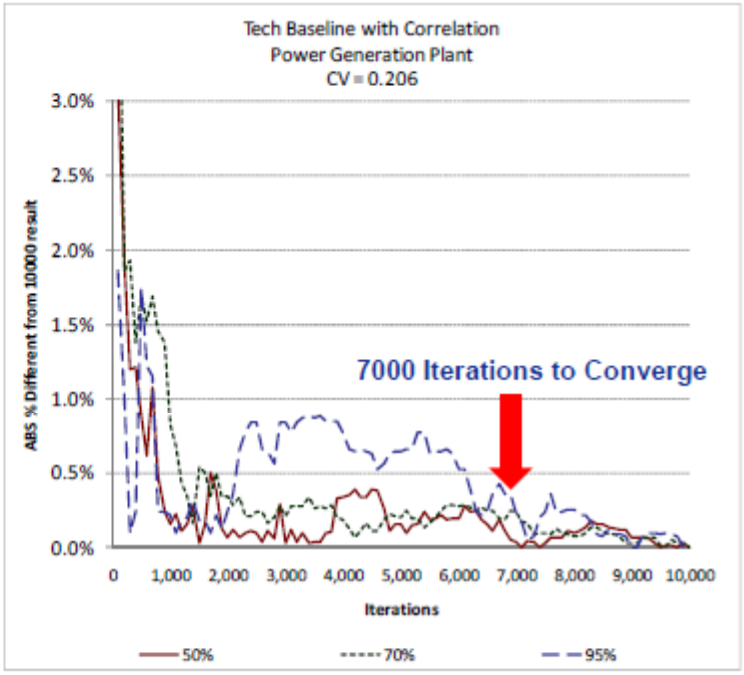
Convergence (cont)

- **Provides guidance on the number of iterations to use in the uncertainty analysis**
  - Measures relative difference of 50, 70, 90 percentile (default, user can choose others) results to those at 10k iterations (or other selected maximum)
  - Applying correlation requires more iterations to converge

Convergence (cont)

- When results are within 0.5% of 10k iterations, convergence is assured as any simulation will yield a 0.5% difference from the results by changing the random seed





Tips & Tricks- Templates

Templates

- **Casual ACEIT users have commented that they continue to use Excel instead because they have created templates that they re-use**
  - Templates feed into specific reports or forms
  - Consistency in the way their reports look
- **The same template can be created in ACE and the versatility of POST allows for customizing reports, charts, etc.**

Templates (cont)

- **Create sections of ACE models that can be useful in multiple models**
  - Specific customer requests (PAUC, APUC, other metrics)
  - Phasing by milestones, dates, etc.
  - Date functions, Matrix functions, etc. can be re-used with minor adjustments
  - POST Reports and Charts can be customized to look and report differently to fit needs

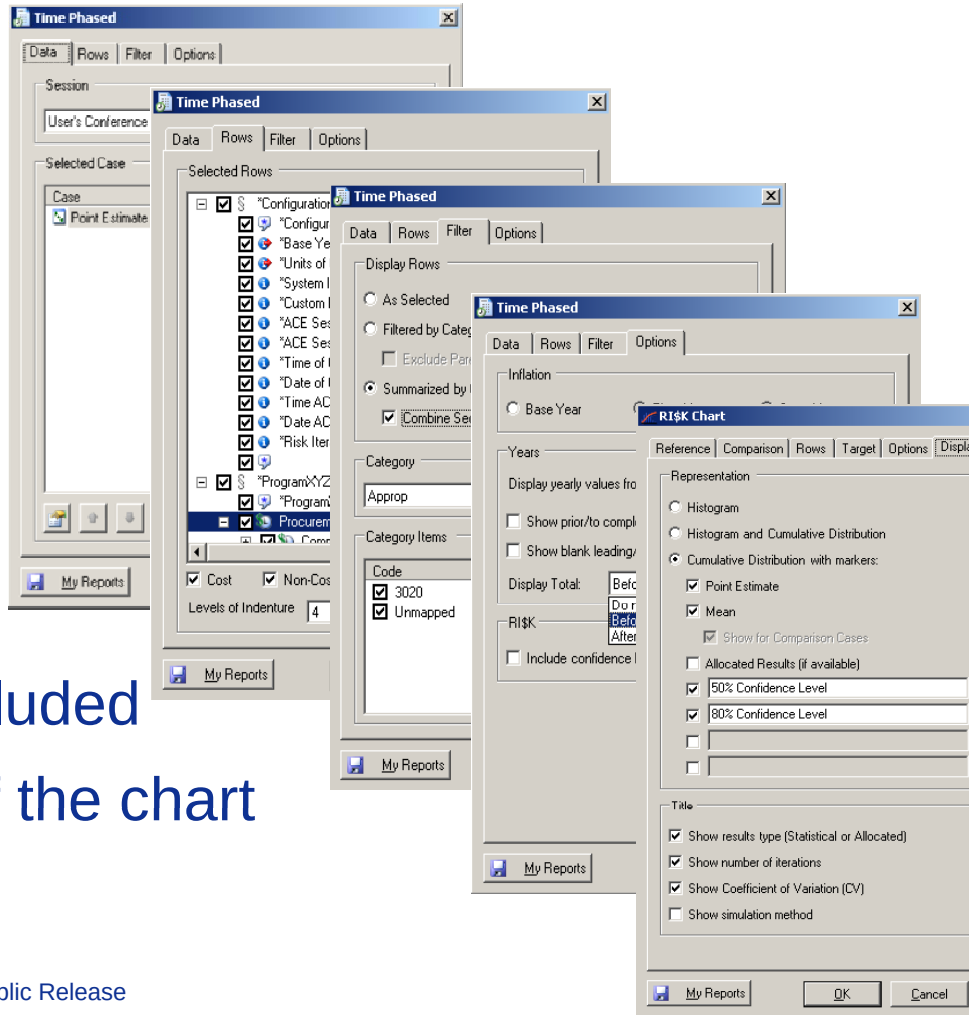
Templates (cont)

- **Templates in ACE models:**
  - Color coded with formatting
  - Add definitions to provide additional insight and documentation

WBS/CES Description	Unique ID	Point Estimate	Phasing Method	Equation / Throughput	Fiscal Year	Units
****Period of Performance Inputs****						
Program Start Date	StartDate	MAR2012 *	C	1Mar2012		
Period of performance (in months)	BasePoP	37.000 *	C	37		
Last date (drop dead finish date) [informational only]	MaxDate	0.000 *	C	*[informational only - not necessary]*		
[F3] Added coverage until FY Funds are available (in months)	stFYCoverage	3.000 *	C	*[F3]* 3		
Period of Performance RISK INPUTS						
Program Start Date Early	LowStartDate	1JAN2012 *		1Jan2012		
Program Start Date Late	HighStartDate	1JUL2012 *		1Jul2012		
Smallest PoP	LowPoP	36.000 *		36		
Largest PoP	HighPoP	40.000 *		40		
Period of Performance Calculations Do Not Change**						
Period of performance integer	PoP	37.000 *	C	RndUp(BasePoP)		
Real Period of performance (in months)	RealPoP	40.000 *	C	PoP + (If(LastFYMonthDifferential<1 ,0,LastFYMonthDifferential))		
Number of max months [only if drop dead date is specified]	MaxMonths	1,346.065 *	C	DATEMONTHDIFF(StartDate, MaxDate)		
REAL Number of Fiscal Years spanned	REALPoPYrQty	4.000 *	C	DATEYR(DATEADD(StartDate, 0, RealPoP - FirstFYCoverage)) -		
Number of Fiscal Years spanned	PoPYrQty	4.000 *	C	DATEYR(DATEADD(StartDate, 0, PoP - FirstFYCoverage)) -		
End Date	EndDate	1JUL2015 *	C	DATEADD(StartDate, 0, RealPoP)		

Templates (cont)

- **POST contains 27 Reports that can be customized and saved**
  - Select 1 or multiple cases
  - Filtered by category
  - BY, TY, SY
  - Risk (un/adjusted)
    - Statistical
    - Allocated
  - Various levels of indenture
  - Specific rows included/excluded
  - Format the look and feel of the chart
    - Colors, fonts, etc.



Summary

Presentation Summary

- **ACEIT and Excel:**
  - Just because you have a nail gun and a drill doesn't mean you won't use your hammer and screwdriver
  - ACEIT has become more versatile, and provides the functionality necessary for just about any estimate
  - Continuous education is valuable to be able to learn and apply tools that will improve the efficiency and effectiveness of cost models

Backup Slides

Error Types

- **FATAL:**
  - ACE found a critical error in your estimate that you must fix before it can generate a result.
  - Typical FATAL problems are improper function, equation, or column syntax.
- **WARNING:**
  - ACE found a problem with your estimate that you should be aware of.
  - For example, you may have specified a methodology on a parent row where it is not appropriate, or you may not have provided a required phasing method.
  - Warnings indicate situations that may or may not affect your results. While an action on your part is not always required, you should examine every WARNING message.
  - ACE still calculates a result if it generates a WARNING.
- **INFO:**
  - ACE found a minor problem with your estimate and will take an appropriate action unless you cancel the calculation.
  - For example, ACE may add a new row to your estimate if it finds a variable that has not been defined.
  - INFO messages typically do not change your results, but you should examine them to be sure.
  - ACE still calculates a result if it generates INFO messages.
- **UNUSED VARIABLES:**
  - ACE found variables with Unique IDs that were not entered any place in the ACE session.

Milestone Date Sections - DateAdd Function

- **When working with flexible schedules there are three main elements to include in the estimate:**
  - Start Date – enter the activity start date
  - Duration – enter the duration of the activity in years, months or days
  - Finish Date – calculated from the start date and duration
- **Use DateAdd() to add Years, Months and/or Days to a date**
- **If decimal values are specified for years or months they will be recognized**
- **Decimal day values will be truncated**

Milestone Date Sections - DateAdd Function

- **Implement the DateAdd function with year, month and/or day durations**
- **Function Syntax:**
  - DateAdd(Date, Year [,Month [,Day [,Truncate]]])
    - Date – Enter the starting date
    - Ways to enter duration
      - For **years** only, enter a variable for the year and remove the month and day parameters (e.g., DateAdd(2007, 5))
      - For **months** only, enter 0 for the year, include a variable for number of months, and remove the day parameter (e.g., (DateAdd(2007,0,13))
      - For **days** only, enter 0 for the year and month, and a enter a variable for the number of days (e.g., DateAdd(2007,0,0,200)
      - You can build durations using combinations of years, months, and days
      - **Truncate** is optional and indicates whether or not the function should operate on fractional years and months, e.g., 1.5 years. By default, the function **does** recognize and add partial years and months.
    - Be sure to remove the square brackets for the optional parameters
- **Most popular implementation uses duration in months**
  - *Example:* DateAdd(Startdate, 0, Duration) – if you leave out the 0 for the year parameter, the month is interpreted as a year

DateOf Function

- **DateOf (Year, [Month], [Day], [Year_Type])**
  - *Year* - This argument is used to reference the year of the date.
  - *[Month]* - Optional argument used to reference the number of months (positive or negative) into the specified year. The first month of a year is specified as the value “1”.
  - *[Day]* - Optional argument used to reference the number of days (positive or negative) into a month. The first day of a month is specified as the value “1”.
  - *[Year_Type]* - This is an optional argument used to reference whether the specified year is a Fiscal Year (0) or a Calendar Year (1).

MatDot Functions

- **MatDot can be used instead of MatColCol or MatColRow. It multiplies multiple columns or rows together.**
  - MatColDot (Num_Rows, Index1,@Var1, Index2,@Var2,…)
    - Indexi is the column index for the vari. When the Indexi = 0 the total @var is used. When Indexi >0 then @vari is the column index of the matrix. When Indexi<0 then the @vari is calculated for each year and the equation should be time phased.
  - MatDot(N_Rows, Type1, Index1, @Var1, Type2, Index2, @Var2,…)
    - The typei argument determines whether @Var is a row or a column vector (0=column, 1=row). The indexi provide the same function as in the MatColDot.

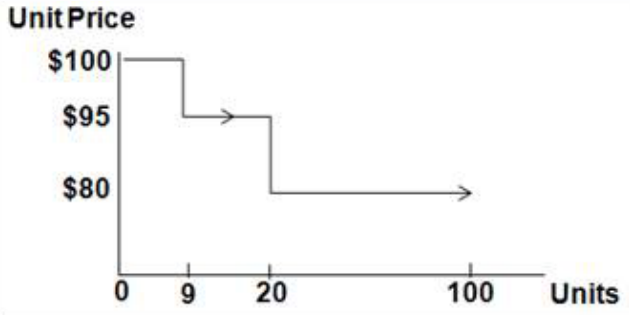
The VLookup Function

- **VLookup Function Syntax:**

VLookup(LookupVal, @DataTable, ColIndex, NumRows)

- *LookupVal* – The value to search in the first yearly column of the DateTable matrix. If the exact value is not found, the value that is closest but not greater than the LookupVal will be used. If the LookupVal is larger than the largest value in the first yearly column of the DataTable matrix, VLookup() returns the largest value.
- *@DataTable* – A table of data (matrix) using the yearly columns to store the data. The values in the first column are the ones searched. As with all ACE matrices, the row address @DataTable is the row preceding the actual matrix.
- *ColIndex* – The column number of the *@DataTable* matrix from which the value must be returned.
- *NumRows* – The total number of rows in the *@DataTable* matrix.

The StepVal Function



- **StepVal Function Syntax:**

StepVal(xval, @x, @f_of_x, num_steps)

- *XVal* – The value or expression for which the step function evaluates.
- *@x* – The array of values defining the highest values for each x-range, that is, where each "step" occurs. (As with all ACE matrices, the row address @x is the row preceding the actual matrix data and is just a marker for the beginning of the matrix.)
- *@f_of_x* – The array of values defining the step function value in the ranges defined by @x, that is, the value at each step.
- *Num_steps* – The number of years with non-zero values defined by @x and @f_of_x

OPCycle Function

- **OpCycle (*Value, StartYear, CycleYears* [*,Multiplier, MaxCycles*])**
  - *Value* - The value to be entered each cycle. This can be a number or a variable, but must be a constant (i.e., phased using C phasing method)
  - *Start Year* - The first year the cycle begins. This may be a decimal value to denote a cycle beginning in the middle of the year (e.g., 2006.5 would be six months into FY 2006)
  - *Cycle Years* - The number of years between each cycle. This may be a decimal value for spans other than years (e.g., 18 months would be 1.5)

OPCycle Function (cont)

- **OpCycle (*Value, StartYear, CycleYears* [,*Multiplier, MaxCycles*])**
  - *Multiplier* - This is an optional parameter. The Multiplier to increase or decrease the value each cycle. The multiplier may be any positive number. A Value of 1.0 means the value will be the same each cycle. 1.0 is the default value if not defined.
  - *Max Cycles* - This is an optional parameter. The maximum number of cycles in the schedule. If excluded, ACE will extend the cycle through the last year of the session.

OPCycle Function (cont)

- **OpCycle**
  - This function is ideal for software maintenance and depot maintenance refresh schedules.
  - The function can be year constrained using the Start Date and Finish Date columns.
  - The function requires the row to be phased with the “F” phasing method.
  - If you want to repeat a yearly throughput (e.g., buy schedule), use the FYRepeat function.

FYRepeat Function

- **FYRepeat (@var, NumTimes, RepeatSize, [FY])**
  - *@var* - This argument refers to the ACE row containing a procurement schedule. The @var syntax identifies an ACE row, where var is the Unique ID of the row (item) to reference
  - *NumTimes* - This argument is the number of times you wish to have the procurement schedule (@var) repeated
  - *RepeatSize* - This argument is the number of years before repeating the schedule again
  - *FY* - This optional argument tells which fiscal year of the repeated schedule to retrieve

FYRepeat Function (cont)

- **FYRepeat**
  - This function can be used in place of a fiscal year argument
  - This function can be used within other functions as part of an expression