User Defined Functions are Not That Scary

Overview

- Why Bother?
- Nuts ‘n Bolts
- Addressing @
- CS in 15 Minutes
- UDF Design Tips
- UDF Toolbox
- Performance Concerns
- Troubleshooting Tips

I always find it more difficult to say the things I mean than the things I don't.

- W. Somerset Maugham

They Scare Me!

- **Fearing the User-Defined Function**
  - The syntax confuses me
  - I'm not using a generalized algebraic equation
  - They are impossible to debug
  - No time to "plan for change"
  - I prefer copy/paste
  - Too late to use one now
  - …

Fear is the path to the Dark Side.

- Yoda

Why? Why Not!

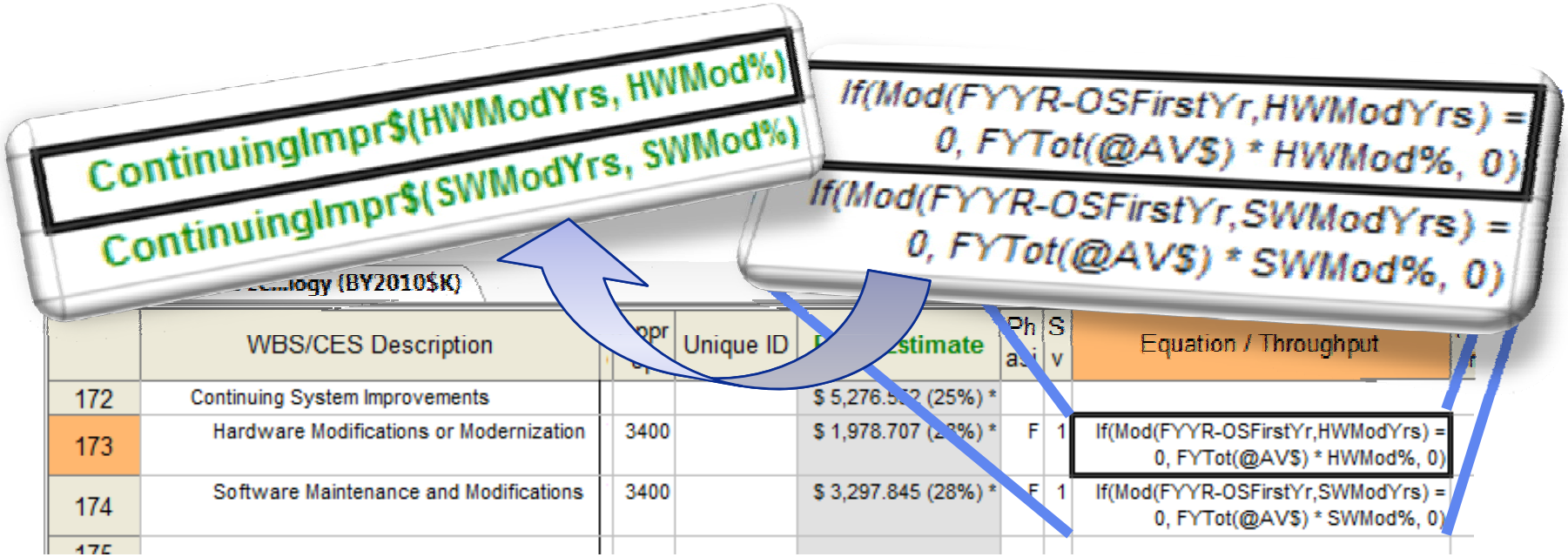
- **Embracing the UDF**
  - Capture & reuse strategies
  - Avoid repeated repetition
  - Isolate & localize complexity
  - Document intentions
  - Facilitate flexibility
  - Earlier is always easier

Small opportunities are often the beginning of great enterprises.

- Demosthenes

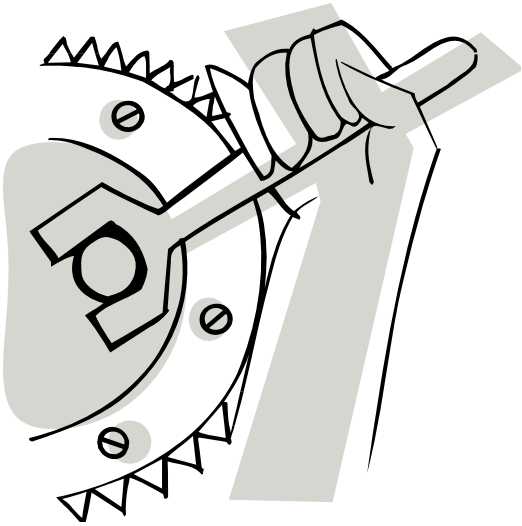
Quick Example

- **Rows below contain a common estimating strategy**
  - To alter the strategy– ***ALL*** rows must be edited
  - If the strategy was **isolated** to a UDF—edit ***ONE*** row
- **Plus, easier to review CER intention with UDF**

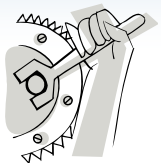


Ace Example File: 07 - Detailed LCC Estimate.aceit

UDF Nuts & Bolts

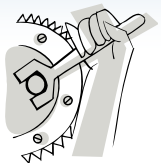


What *IS* a UDF?



- **You create a User-Defined Function (UDF) to:**
  - Centralize a repeated calculation
  - Separate control from cost calculations
  - Hide details so that changes are easier
- **A UDF is *defined* on a single row in your session.**
  - But, a UDF row is ***never*** evaluated.
  - Instead, it is evaluated *inside **other** rows'* equations.
- **A UDF behaves just like a Built-In ACE function**
  - Arguments and result are in ***row's units*** (wrapped)
  - Common Error: Assuming UDF is in Session Units

UDF Declaration



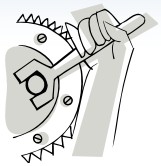
- **A UDF Consists of Four (4) Parts** *[in 3 columns]*:
  - Description—*to distinguish it from a comment row*
  - Unique ID—*must be unique to whole session*
  - Argument List—*values used in its equation*
  - Equation—*the math used to produce a result*

Only three columns are active on a UDF row

	WBS/CES Description	Unique ID	Point Estimate	h	Equation / Throughput
20	UDF - Fuel Costs	Fuel$(Miles, MPG, PerGal$)			Miles * PerGal$ / MPG
21	UDF: Trucate unless close	Truncate(uNum)			RndDn(uNum + 0.05)

Two parts to UDF declaration – name & list of arguments

UDF Evaluation



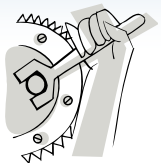
- **Think "Inline Substitution"** *(almost)*
  - You can "insert" a UDF into equation and get the same result
  - This metaphor helps visualize context of UDF calculation
  - It is important to note that numbers are substituted—*not text*

Rows 37 & 38 produce same result

	WBS/CES Description	Approp	Unique ID	Point Estimate	P h	Equation / Throughput	F
35	***UDF: Data Proc***		***DP(W,S)***			***2 * W^0.5 * 1.5^S***	
36							
37	Using DP() w/ Num			50.000 *	C	30 + DP(100,0)	
38	Expanded w/ Num			50.000 *	C	30 + (2 * (100)^0.5 * 1.5^(0))	

Example from ACE Help topic "User Defined Functions"

Evaluation Walkthrough



	WBS/CES Description	Ph	Equation / Throughput
40	Using DP() w/ Var	C	15 + DP(WT, SEA)

1) Resolve Args

WT = 88

SEA=1

40	Using DP() w/ Var	C	15 + DP(88, 1)

2) Sub Args in UDF

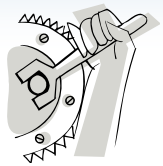
DP(W=88, S=1) = 2 * W^0.5 * 1.5^S

= 2 * 88^0.5 * 1.5^1

3) Sub UDF in Equ

40	Using DP() w/ Var	C	15 + (2 * 88^0.5 * 1.5^1)

ID Visibility



- **UDF with same name *hides* built-in function**
  - Useful if you don't like how ACE implemented a function
  - **Not recommended** due to ambiguity and confusion it causes

WBS/CES Description	Unique ID	Point Estimate	Ph	Equation / Throughput
My Rounding UDF	RndDn(a)			Rnd(a - 0.05)

- **UDF *argument* with same name hides Unique ID**
  - A necessary evil--Beware of the confusion that may arise
  - Sharing names among UDFs is a good thing (limited scope)

Hidden

	WBS/CES Description	Unique ID	Point Estimate	Ph	Equation / Throughput
18	Miles travelled	Miles	2,000.000 *	C	2000
20	UDF - Fuel Costs	Fuel$(Miles, MPG, Gal$)			Miles * Gal$ / MPG

Hider

'Local'

Addressing @

(How to impersonate a row ID)

Problem: Built-In @ Arguments

- **Some ACE functions need a row address: *FycMax(@Row)***
- **Yet, a UDF translates its arguments to numbers**
  - And you cannot apply an "@" operation to a number

ACE Error: "@ applied to number"

ACE thinks "Row" is a number

	WBS/CES Description	Approp	ique ID	Point Estimate	P h	Equation / Thro	t
48	Total		Total	115.000 *	F	FY	CT
49							
50	Improper @ Declare		Wrong(Row,Val)	*MTH7056		Val * FYCMax(@Row)	
51	Improper Usage				C	Wrong(Total, 1.2)	
52	Still Improper Usage				C	Wrong(@Total, 1.2)	

Inline Substitution:
1.2 * FYCMax(@115.0)

Inline Substitution:
1.2 * FYCMax(@48.0)

Solution: UDF @ Arguments

- **Define argument to accept a row address w/ @ prefix**
  - This is a number *specially marked* to access another row's results
- **The argument name is an *alias* for the row number passed in**
  - It's like creating a temporary row ID that is used only inside of UDF

"@" tells ACE the "Row" argument behaves like a Unique ID of a row.

	WBS/CES Description	Appr	Unique ID	Point Estimate	Ph	Equation / Throughput
48	Total		Total	115.000 *	F	FYFACT
49						
54	Proper @ Declare		Right(@Row, Val)			Val * FYCMax(@Row)
55	Proper Usage			36.000 *	C	Right(@Total, 1.2)

Inline Substitution:
1.2 * FYCMax(@Total)

@ Argument Example

	WBS/CES Description	Unique ID	Point Estimate	Ph	Equation / Throughput
48	Total	Total	115.000 *	F	FYFACT
57	Use @ UDF		235.000 *	F	Max$IfLater(@Total)
58	UDF - Out years receive max	Max$IfLater(@C$$$)			If(FYYR<=C$$$.LastYr, C$$$, FYCMax(@C$$$))

If(FYYR<=C$$$.LastYr, C$$$, FYCMax(@C$$$))

"Dot" Attribute

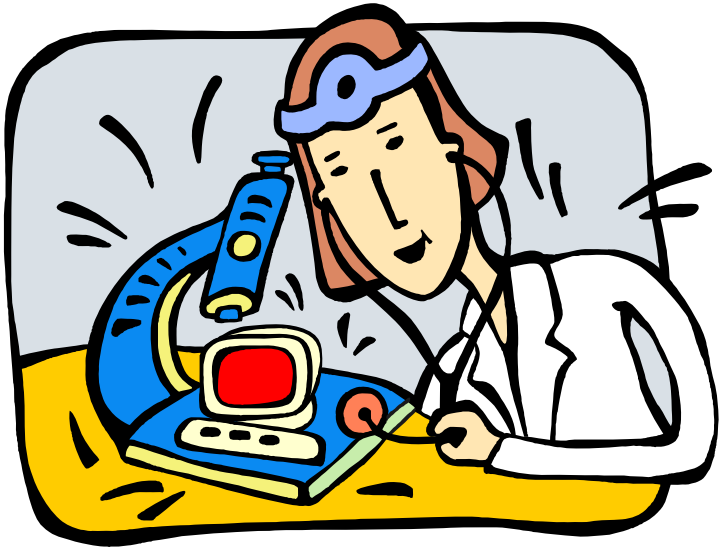
"Value" of C$$$

"@" Reference

07 - Detailed LCC...ology (BY2010$K) | Basic UDF Fuel C...ology (BY2009$K) | Basic UDF Fuel Co... Point Estimate)

	Cost Element	Approp	Total	FY 2009	FY 2010	FY 2011	FY 2012	FY 2013	FY 2014	FY 2015	FY 2016	FY 2017	FY 2018	FY 2019
1	Total		115.000		10.000	20.000	30.000	20.000	25.000	10.000				
10	Use @ UDF		235.000		10.000	20.000	30.000	20.000	25.000	10.000	30.000	30.000	30.000	30.000
11	UDF - Out years receive max													

Comp Sci in 15 Minutes



What I learned in CS

- **Change is Inevitable–*especially when assured otherwise***
- **Refine through Iteration—*nothing is ever complete***
- **Hide Details—*expose intent and expectations***
- **Test Early and Often—*hope springs eternal bugs***
- ***Determine, Capture and Isolate Strategies***

If I had asked people what they wanted, they would have said faster horses.

- Henry Ford

Hiding "How"... *Abstraction*

- **Layer details of a strategy under an interface**
  - ***Shows what*** is expected (e.g. argument, context)
    - The row populates the UDF arguments
  - ***Hides how*** it is implemented (i.e. math)
    - The UDF equation implements the math "behind the scenes"

Travel$(Distance, FuelRate, MPG)

How Travel$ is calculated is hidden away on another row

- ***Advantages***:
  - Change underlying calculation at any time ('cause it's hidden)
  - Use UDF instead of copying, decoding & modifying its math
  - Verifying a CER's ***intent*** just got a lot easier

Isolate and Refine… *Iteration*

- **Introduce UDFs earlier rather than later**
  - Your UDF does **not** have to be a finished product
  - You can always come back to refine your thinking
  - In this way, you only have **one place** to refine (or repair)

<u>My Travel Cost UDF:</u>

~~Distance * 0.35~~

~~Distance * FuelRate / MPG~~

Dist * (MaintRate + FuelRate / MPG)

Building Blocks… *Reusability*

- **Build Building Blocks of UDFs**
  - Even a simple build-up easier to interpret as a UDF
  - *Ex: Suppose "Area" is common in a session's build-ups:*

7.1 * 3.55 * 7.25	VS	Material$(7.1, Area(3.55, 7.25))
.4*(2.5*7.25)^.5		Labor$(0.4, Area(2.5, 7.25))

- **UDFs are much easier to borrow than CERs**
  - No need to hunt through equation to replace variables
  - Hint: Check out "Section Templates" in ACE help

Bookkeeping… *Encapsulation*

- **Separate decision-control from WBS/CES**
  - IF() and SEL() are best stashed elsewhere
  - For instance…
    - selecting among several values,
    - filtering values based on type,
    - applying adjustments (nudges, fudges or errors),
    - boundary tests and corrections
- **Watch for patterns developing in WBS**
  - Ask if the row has a need to know
  - If not, decouple decision-control from cost calculation

Bookkeeping Example

- **User wanted zeroes to appear in phased reports**
  - Every Row in WBS contained following logic:

    IF([CER]>0, [CER], 0.0001)

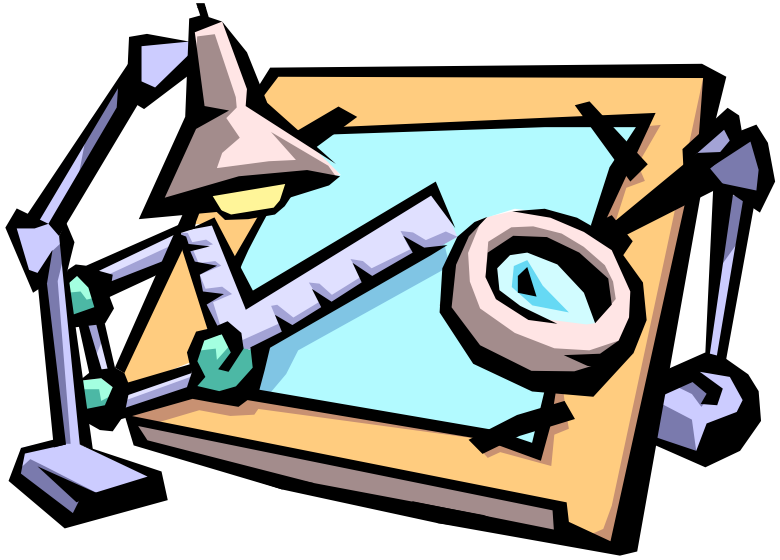
  - But this logic isn't "row specific" -- unimportant to row
  - Cannot "turn off" behavior without editing every equation in WBS
- **Recommend abstraction/encapsulation:**
  - Each row's CER becomes…

    ShowZero([CER])

  - Row *requests* zeroes in report but doesn't *control* report setting.

    UDF: ShowZero(X) **=** IF(X>0,X,IF(Hide, 0, 0.0001))

UDF Design Tips



Hunting For Repetition

- **Don't try to guess what you need**
  - Let the session structure emerge first
  - But watch for repetition—tendency to copy/paste/edit rows
  - Introduce UDF on next refinement iteration
- **Judicious pattern matching**
  - Identify the calculation strategy that rows have in common
    - Not just the arithmetic symbols in common
    - Not just a list of different unique IDs
- **You need to identify 3 things:**
  1) Which part of the row's equation is in common
  2) Which parts vary from row to row
  3) Which values to pass in instead of calculate internally

Example of 3 Parts

- **The two rows below have obvious similarities**
- **The whole CER can be converted to a UDF**
  - no row-specific fringe to leave behind

07 - Detailed LC...logy (BY2010$K)

	WBS/CES Description	Appr op	Unique ID	Point Estimate	Ph asi	S v	Equation / Throughput	Fi Y
172	Continuing System Improvements			$ 5,276.552 (25%) *				
173	Hardware Modifications or Modernization	3400		$ 1,978.707 (28%) *	F	1	If(Mod(FYYR-OSFirstYr,HWModYrs) = 0, FYTot(@AV$) * HWMod%, 0)	
174	Software Maintenance and Modifications	3400		$ 3,297.845 (28%) *	F	1	If(Mod(FYYR-OSFirstYr,SWModYrs) = 0, FYTot(@AV$) * SWMod%, 0)	

- **2 parts vary from row to row**
  - HWModYrs & SWModYrs
  - HWMod% & SWMod%

```
If(Mod(FYYR-OSFirstYr,HWModYrs) =
    0, FYTot(@AV$) * HWMod%, 0)
If(Mod(FYYR-OSFirstYr,SWModYrs) =
    0, FYTot(@AV$) * SWMod%, 0)
```

- **But is that our strategy?**

Example of 3 Parts (cont)

- **There are two equally viable "strategies" here.**
  - One expects the % of AV$, the other the actual cost:

ContImpr$(ModYrs, **Mod$**) **VS** ContImpr$(ModYrs, **Mod%**)

- **It isn't always in your best interest to pass in the rudimentary variables and do all the calculation in the UDF.**
- **Which to use depends on where the session is heading.**
  - Passing a cost is more general but requires an intermediate calc.
  - If AV$ is always used, the intermediate calc clutters the row.

    Usage: ContImpr$(SWModYrs, **AV$.FYTot * SWMod%)**
  - Passing a Mod% encapsulates the intermediate calc in the UDF.
  - But the UDF is only useful when improvement depends on AV$.

    Usage: ContImpr$(SWModYrs, **SWMod%)**

Example 2 - Buildup

- **Remember the "Area()" building block?**
  - Did we really need Area() UDF?
  - Is it our strategy?
  - Do we plan to use it elsewhere?
  - More sense to calculate directly?
  - Should we pass dimensions into UDF?

0.4*(2.5*7.25)^.5

Usage: Labor$(0.4, Area(2.5, 7.25))

Usage: Labor$(0.4, 2.5 * 7.25)

Usage: Labor$(0.4, 2.5, 7.25)

UDF Naming Tips

- **Describe the result in the UDF name**
  - This is called ***self-documenting*** and is a cool CS technique
  - Avoid using names that differ by only a letter or two

Unique ID	Equation / Throughput
OP(A,B,C)	A * Max(1.0, (A/B)^(1+C))
PenalizeOverrun(Cost, Thresh$, PenAdj)	Cost * Max(1.0, (Cost/Thresh$)^(1+PenAdj))

- **Use descriptive words *(or abbrvs)* for argument names**
  - Names are local, so you can use short names
  - Avoid using 1-2 letter names for arguments
  - Include expected units in name to clarify how to call UDF

Sin(X) → Sin(Angle) → Sin(Radians)

UDF Toolbox

Access "Dotted" Value

- **Problem: Can't access dotted value**
  - Syntax won't let you get to DEC with row offset

(@Row+X).aStartDate + Duration

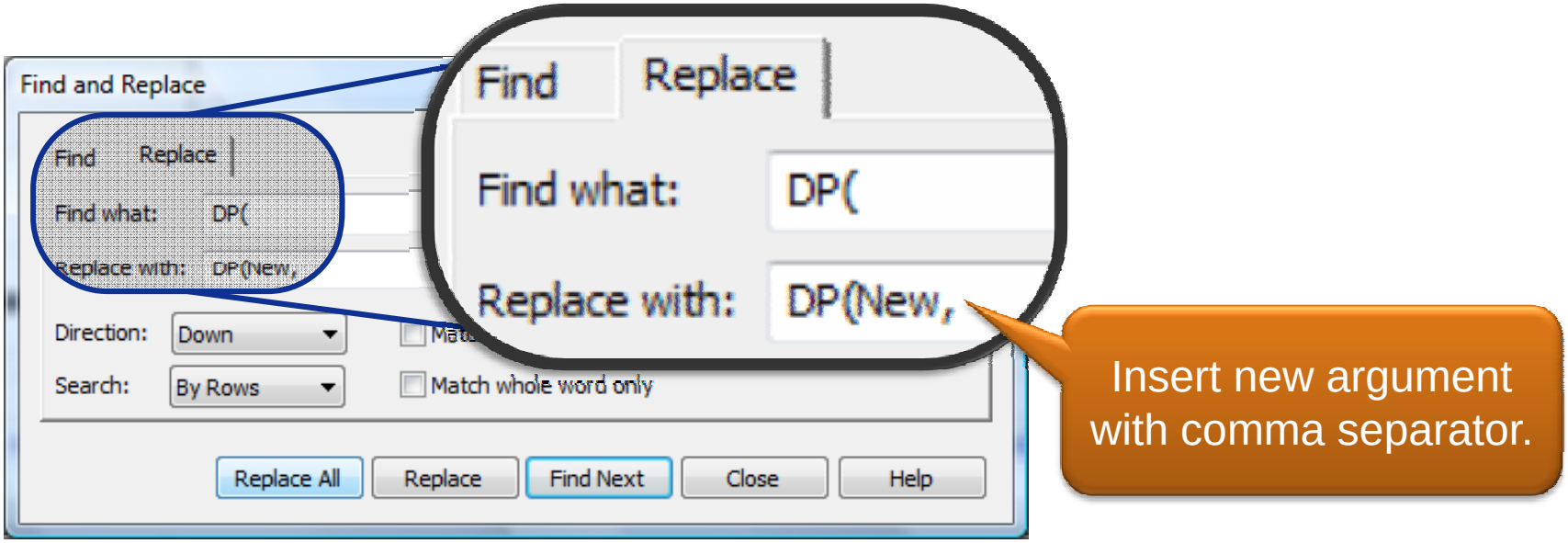
- **Solution: UDF that takes row and returns value:**
  - Note: You would need one UDF for each DEC

WBS/CES Description	Unique ID	Equation / Throughput
UDF - Start Date at Row	StartAt(@Row)	Row.aStartDate

	WBS/CES Description	Unique ID	Equation / Throughput	Point Estimat	Ph				Offset	Start Date	Finish Date
62	Usage Example		1	4.000 *	F				2	StartAt(@Tbl+Offset)	aStartDate +1000
63	Table of values	Tbl		6.000 *							
64	a		1	3.000 *	F					01OCT2011	01OCT2013
65	b		1	3.000 *	F					01SEP2012	01SEP2014

Adding an Argument

- **What if you realize that you need another variable passed into your UDF?**
  - Add new name to the ***front*** of argument list
    - Replace UDF name and open parentheses with default "placeholder" value as shown below.
    - Don't forget the separating comma!



DEC as Backdoor Argument

- **Use DECs as "backdoor" arguments**
  - Reduces clutter in the "Equ/Thrupt" cell
  - Fewer arguments to declare and pass into UDF
  - Useful for flags, type arguments, and WBS row offsets
  - Opens the way for category filtering using **SUMIF()**

	WBS/CES Description	Unique ID	Equation / Throughput	Rd (!) Radius	Len (!) Length	Wth (!) Width	Point Estimate
73	Wood $/sf	Wood$sf	3.0				3.0 *
74	Iron $/sf	Iron$sf	5.0				5.0 *
76	**Wood Tube**		**Tube(Wood$sf)**	**2**	**4**		**150.8 ***
77	**Iron Tube**		**Tube(Iron$sf)**	**1**	**8**		**251.3 ***
78	**Wood Board**		**Rect(Wood$sf)**		**3**	**4**	**36.0 ***
79	**Iron Plate**		**Rect(Iron$sf)**		**3**	**5**	**75.0 ***
81	**UDF - tube**	**Tube(sf$)**	**sf$*6.283*Rd*Len**				
82	**UDF - Rect**	**Rect(sf$)**	**sf$*Wth*Len**				

Performance Concerns

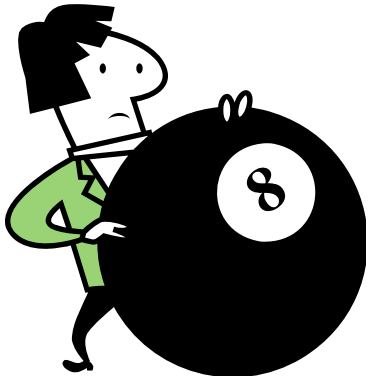
UDF Performance

- **Yes, UDF is slower than direct evaluation**
- **Count on row calc time to roughly double *(WAG)***
  - That means if 20% of rows use a UDF, your session will take 20% longer to calculate.
- **Yes, DEC is slower than no DEC**
- **Count on row calc time to roughly double**
  - That means if 20% of rows use DECs, your session will take 20% longer to calculate.
- **Higher math: Using both UDFs and DECs**
  - If 20% of rows use both, calculation takes 60% longer!

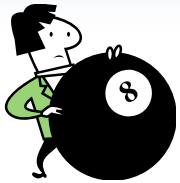
Performance Tuning

- **Some time savings found when…**
  - UDF has fewer arguments
  - UDF uses short argument names *(dissimilar prefix)*
  - Intermediate calculations performed as argument
    - *E.g., MyFunc(X*B^E, 1.2/B)*
    - AND when argument used multiple times in UDF
- **For "F" method rows, consider using Start/Finish years.**
- **For RI$K calculations…**
  - Default to small number of iterations for "Draft" reports.
  - Set large number of iterations in "Final" reports.

Troubleshooting Tips



Tips for Testing

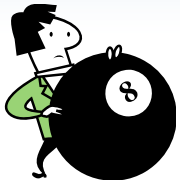


- **The first rule in testing is…**
  - Know the answer ***before*** you run the test.
- **Start by assuming that you did something wrong.**
  - If you did it right, it would work.
  - Mistakes hide well within one's certainty.
- **Look for stupid stuff first.**
  - Use Traceback dialog or hover tips to verify variable descriptions.
  - Make sure "@" usage matches UDF declaration.

Computers have the annoying habit of doing exactly what they are told.

- CS Proverb

Tips for Testing



- **Check assumptions of UDF equation.**
  - Expect to be used on “C” or “F” method? Costs in certain units?
- **Work from inside out.**
  - Find a place where you get known, desired behavior.
  - Then, work outwards until expectation fails.
- **Isolate in separate, small session file.**
  - Get away from the clutter of a complex session.
  - Makes it easier to dissect UDF without breaking calculation.
- **Beware of RI$K.**
  - Does distribution approach zero? Can value become negative?
- **Remember UDF evaluation sequence:**
  - Resolve argument values, insert values into UDF, insert UDF into row/cell equation.

In Review

- **UDFs aren't just for math majors**
- **Use UDFs to centralize cost & control strategies**
  - CS concepts of "Abstraction" and "Reuse"
- **Remember "inline substitution" metaphor**
  - UDF takes on context of row's (cell's) using it
- **Test UDFs by isolating them—go from inside out**
- **Don't worry too much about performance**
- ***The more you use them, the easier they become***