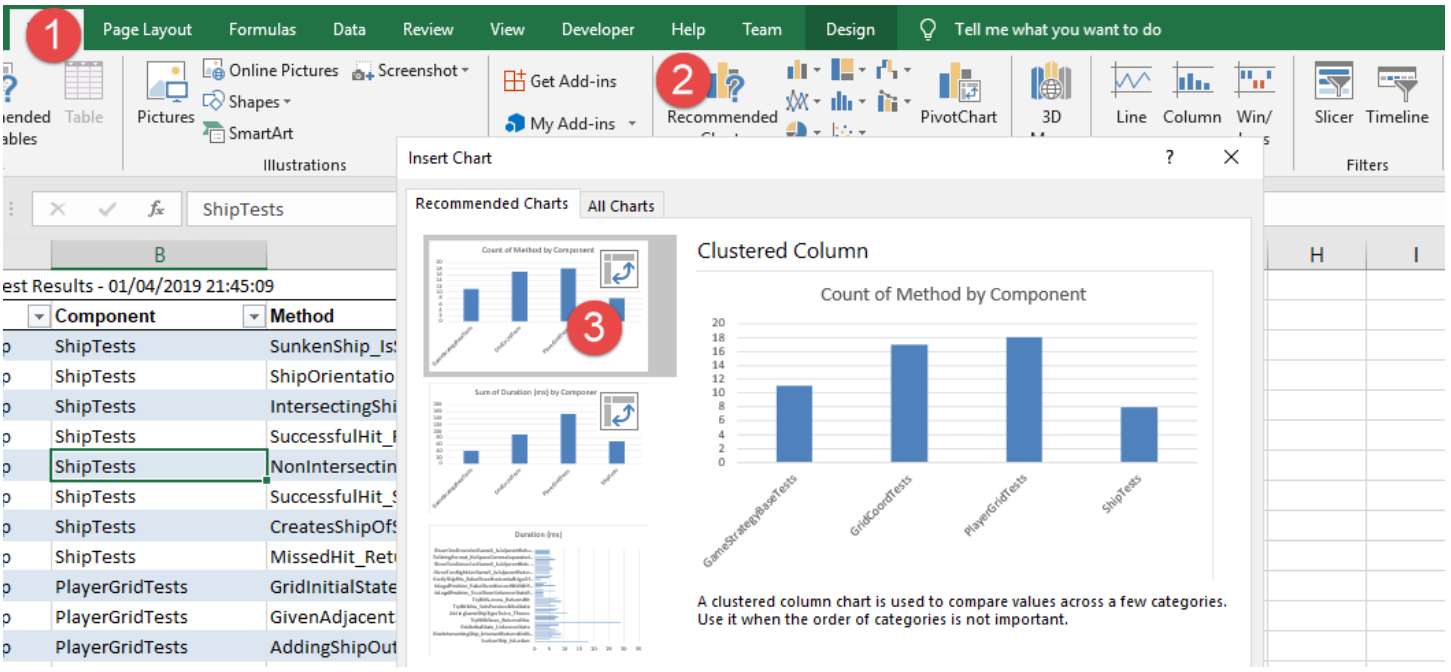
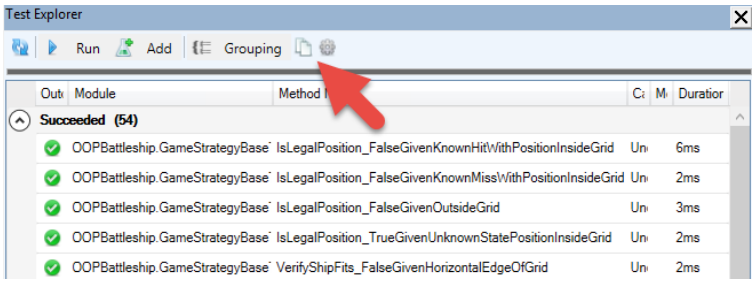


Code Insights with Rubberduck + Excel

Posted on January 5, 2019 by Rubberduck VBA

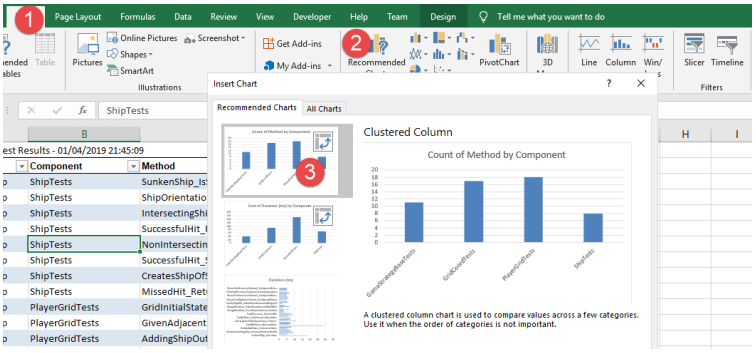
You're writing a rather large VBA/VB6 project, and you're starting to have a sizable amount of passing unit tests. Did you know you can copy the test results to the clipboard with a single click?



…and then paste them onto a new worksheet and turn it into a data table:

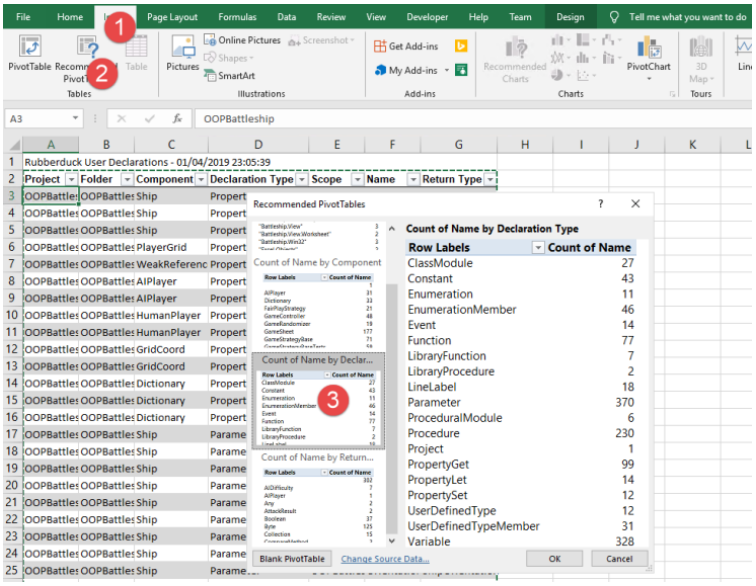
	A	B	C	D	E
1	Rubberduck Test Results - 01/04/2019 21:45:09				
2	Project	Component	Method	Outcome	Duration (ms)
3	OOPBattleship	ShipTests	SunkenShip_IsSunken	Succeeded	18
4	OOPBattleship	ShipTests	ShipOrientationInvalidEnumValue_Throws	Succeeded	5
5	OOPBattleship	ShipTests	IntersectingShip_IntersectReturnsGridCoord	Succeeded	9
6	OOPBattleship	ShipTests	SuccessfulHit_ReturnsTrue	Succeeded	6
7	OOPBattleship	ShipTests	NonIntersectingShip_IntersectReturnsNothing	Succeeded	9
8	OOPBattleship	ShipTests	SuccessfulHit_SetsStateToTrue	Succeeded	7
9	OOPBattleship	ShipTests	CreatesShipOfSpecifiedType	Succeeded	8
10	OOPBattleship	ShipTests	MissedHit_ReturnsFalse	Succeeded	7
11	OOPBattleship	PlayerGridTests	GridInitialState_UnknownState	Succeeded	5

If you're not sure what to do next, you can even let Excel give you ideas – you'll find the *Recommended Charts* button under the *Insert* Ribbon tab:

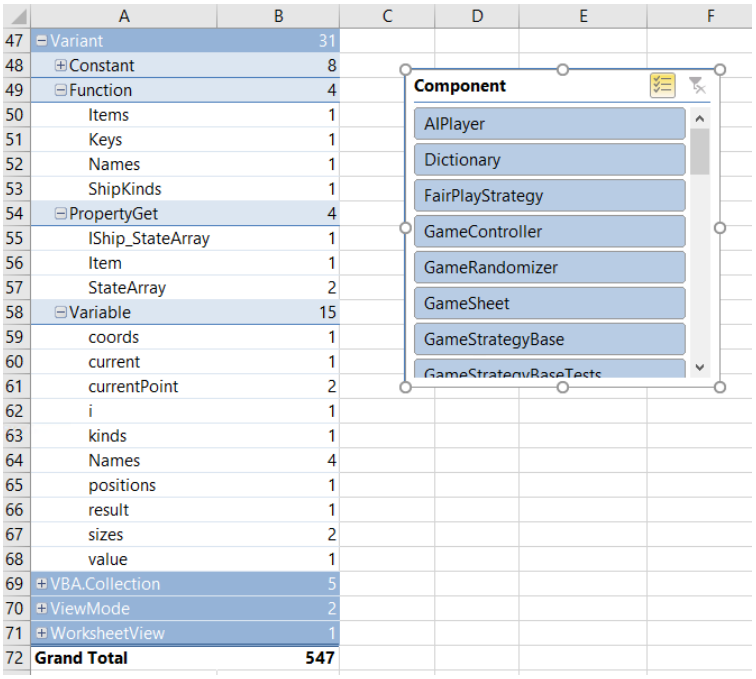


With the *count of method by component* chart, we can see what test modules have more test methods; the *sum of duration by component* chart can show us which test modules take the longer to execute – or we could average it across *test categories*, or archive test results and later aggregate them… and then use this data to performance-profile problematic test scenarios.

Similarly, the "Copy to Clipboard" button from the *Code Explorer* can be used to export a table into Excel, and using the *recommended pivot tables* feature, we can get a detailed breakdown of the project – for example *count of names by declaration type* creates a pivot table that lists all Rubberduck declaration types, so you can easily know how many line labels your project has, or how many `Declare Function` imports are used:



With a little bit of filtering and creativity, we can regroup all *Constant*, *Function*, *PropertyGet* and *Variable* declarations by return type, and easily identify, say, everything that returns a `Variant`:



	A	B
47	⊟Variant	31
48	⊞Constant	8
49	⊟Function	4
50	Items	1
51	Keys	1
52	Names	1
53	ShipKinds	1
54	⊟PropertyGet	4
55	IShip_StateArray	1
56	Item	1
57	StateArray	2
58	⊟Variable	15
59	coords	1
60	current	1
61	currentPoint	2
62	i	1
63	kinds	1
64	Names	4
65	positions	1
66	result	1
67	sizes	2
68	value	1
69	⊞VBA.Collection	5
70	⊞ViewMode	2
71	⊞WorksheetView	1
72	**Grand Total**	**547**

The possibilities are practically endless: the data could be timestamped and exported to some Access or SQL Server database, to feed some dashboard or report showing how a project grows over time.

How would you analyze your VBA projects? What *code metrics* would you like to be able to review and pivot like this? Share your ideas, or implement them, and send a pull request our way!

Posted in excel, ide, OOP, open-source, rubberduck, tutorials, unit-testing, vb6, vba

Published by Rubberduck VBA

I'm Mathieu Guindon (Microsoft MVP Office Apps & Services, 2018), you may have known me as "Mat's Mug" on Stack Overflow and Code Review Stack Exchange. I manage the Rubberduck open-source project, whose goal is to bring the Visual Basic Editor (VBE) - VBA's IDE - into the 21st century, by providing features modern IDE's provide. *View all posts by Rubberduck VBA*