

Test Automation and the challenges of Complex Data

```
0100 0011 0110 1111 0110 1110 0111 0011 0110 1001 0110 0100 0110 0101  
0111 0010 0010 0000 0110 1001 0111 0100 0010 0000 0111 0000 0111 0101  
0111 0010 0110 0101 0010 0000 0110 1010 0110 1111 0111 1001 0010 0000  
0111 0111 0110 1000 0110 0101 0110 1110 0010 0000 0111 1001 0110 1111  
0111 0101 0010 0000 0110 0110 0110 0001 0110 0011 0110 0101 0010 0000  
0111 0100 0111 0010 0110 1001 0110 0001 0110 1100 0111 0011 0010 0000  
0110 1111 0110 0110 0010 0000 0110 1101 0110 0001 0110 1110 0111 1001  
0010 0000 0110 1011 0110 1001 0110 1110 0110 0100 0111 0011 0010 1110
```

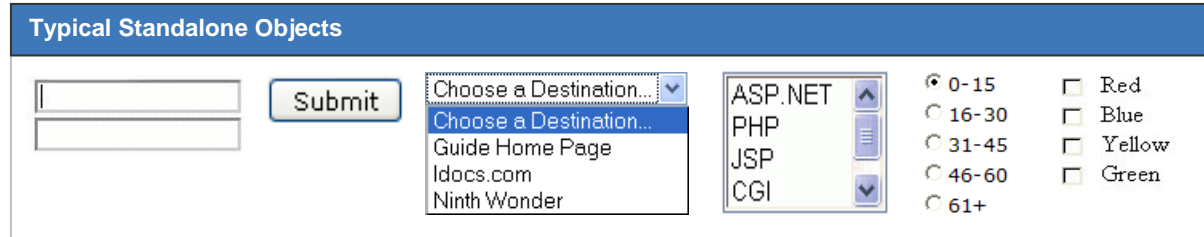
A real world approach to solving tough automation realities

Written by Greg Paskal

Test Automation and the challenges of Complex Data

Overview

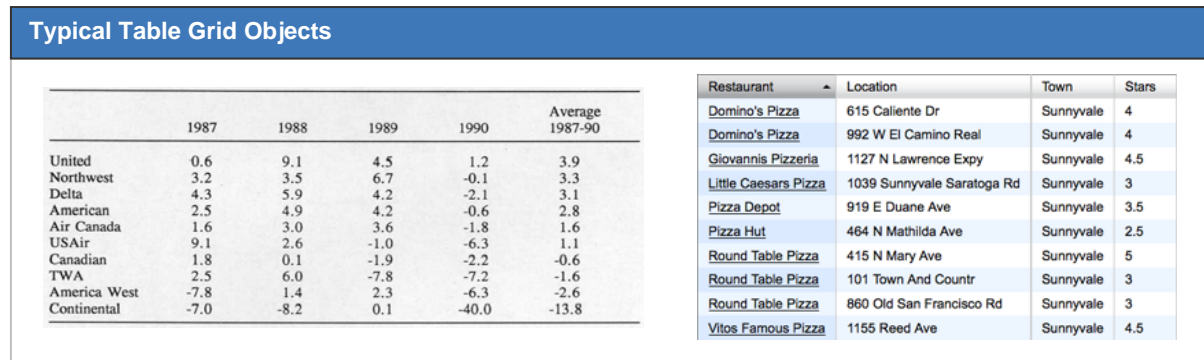
Test Automation engineers regularly encounter a variety of objects that require interaction and verification. In most cases, these objects are standalone such as an Edit Fields, Buttons or Drop-downs.



Some of the more common objects encountered when automating applications.

There is another type of object, made up of groups of standalone objects, which Automators encounter and can be presented in a variety of ways. I refer to these objects as a Table Grid Objects and they typically contain multiple rows and columns of information, similar to a spreadsheet.

This article will discuss an approach to working with Table Grid Objects and the information within them. I refer to this approach as Complex Data Process due to the complex nature of interacting with these Table Grid Objects and the data contained within them.



Common Table Grid Objects such as these present data in a classic row and column format.

Background

While developing test automation for an application in 2009, I encountered a challenge, i.e., how to verify a large volume of data returned in a web accessible dashboard? By utilizing the automation tool's built-in Object Spy, I could see the individual objects that made up the dashboard. Unfortunately though, I could not find a unique identifier for each row or column. I eventually came to the conclusion that if I could identify at least the columns of data and leverage index numbers to navigate to the specific rows, then I would be able to traverse through the landscape of data, verifying as I went along.

What resulted from this initial effort was an elaborate, nested, series of loops, traversing through data, comparing it with expected results and recording the final outcome. The solution worked, but was complicated and difficult to follow. The initial challenge was solved but the complexity of the solution along with an effective way to reuse it was not yet in sight.

A second project came up months later with an even more complex dashboard containing thousands of individual data points stored in multiple Table Grid Objects. Now challenged with some of the most complex validation encountered in my test automation career, I came to the conclusion that there needed to be a better approach to solving this problem, one that was logical and straight forward.

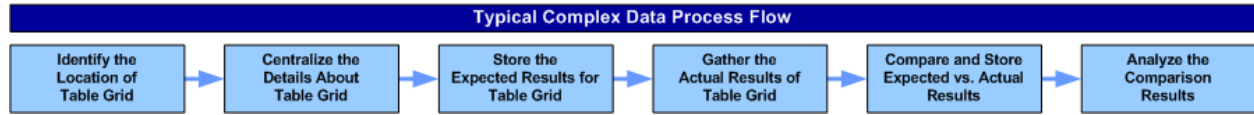
I decided to reverse engineer the solution built for the earlier project, clearly documenting each step of the process so that it could be better understood. The results of this effort were invaluable, as I added a more logical naming strategy to variables and functions; the Complex Data Process was born. Further refinements over the following months revealed how greater efficiency could be realized, resulting in faster execution times.

Test Automation and the challenges of Complex Data

Breaking Down the Complex Data Process

The Complex Data Process is a strategy that is implemented through test automation technologies. Programmatic code functions help implement the process of breaking down and analyzing the data into small, manageable tasks. This creates opportunities to refine and improve it over time. Gains are also realized in reusability, leading to a greater return on investment (ROI).

The Complex Data Process translates into the Complex Data Array, the container that holds everything together in one object. Below is a typical application of this process.



- **Identify the Location of Table Grid** - Foundational to the Complex Data Process is the ability to consistently and reliably work with the Table Grid Object. This gets more complicated when multiple Table Grid Objects exist on a single page (common with dashboards). The cornerstone to solving this problem is the Table Index Key (TIK), discussed later in this paper.
- **Centralize the Details About Table Grid** - Values such as the TIK are centrally stored in an array called the Complex Data Array allowing programmatic functions, access to the required information to locate the Table Grid Object itself. Other, details such as column titles and test candidate flags are also stored in this same centralized Complex Data Array.

Centralizing Important Details	
Data Details	Validation Candidate
CD(0,0,0) = Average Annual Wage Report (Name)	CD(1,0,1) = TIK = 5,2 (TIK)
CD(0,0,0) = Average Annual Wage Report	CD(0,1,1) = TC = TRUE (TC)
CD(0,1,0) = State abbreviation	CD(0,2,1) = TC = TRUE (TC)
CD(0,2,0) = State	CD(0,3,1) = TC = TRUE (TC)
CD(0,3,0) = Average Annual Wage	

Looking inside the Complex Data Array, the TIK value (5,2) along with Column Titles (State abbreviation, State, Average Annual Wage) and Test Candidate Flags (TC=True) can be seen.

- **Store the Expected Results** - A variety of methods can be leveraged to gather Expected Results test data. From manually coding them into the Complex Data Array to reading them in from a data file or database. Whichever method best fits the project, the Complex Data Array provides the centralized location to store this information.

Expected Results Stored Within the Complex Data Array
<pre> ===== Expected Results ===== CD(1,0,2) = 5 (RC) CD(0,1,2) = 3 (CC) CD(1,x,2) = (1)AL (2)Alabama (3)\$40,960 CD(2,x,2) = (1)AK (2)Alaska (3)\$54,040 CD(3,x,2) = (1)AZ (2)Arizona (3)\$37,960 CD(4,x,2) = (1)AR (2)Arkansas (3)\$41,530 CD(5,x,2) = (1)CA (2)California (3)\$58,240 </pre>

These Expected Results should be gathered in the most appropriate manner to the project and application under test (AUT).

Test Automation and the challenges of Complex Data

- **Gather the Actual Results of the Table Grid** - The actual result from the Table Grid Object, known as the Validation Candidate, are gathered by the Complex Data supporting functions. These functions will utilize the TIK stored earlier in the process. The Complex Data Array provides the necessary storage for this gathered data.



No single function will likely solve every situation for gathering Validation Candidate data. (Additional solutions are anticipated)

- **Compare and Store Expected vs. Actual Results** – Once all the test data is gathered it will be compared utilizing the Complex Data supporting functions. The results of these comparisons are recorded directly into the Complex Data Array keeping the test data and all results in one place.

Results of the Comparison
<pre>===== Validation Results =====</pre>
<pre>CD(0,0,3) = Pass (TOR) CD(0,1,3) = 15 (TOV) CD(0,2,3) = 1297412479 (ETS) CD(0,3,3) = 1297412480 (ETC) CD(0,4,3) = 00:00:01 (ETE) CD(1,0,3) = Pass (RR) CD(2,0,3) = Pass (RR) CD(3,0,3) = Pass (RR) CD(4,0,3) = Pass (RR) CD(5,0,3) = Pass (RR) CD(1,x,3) = (1)Pass - "AL" = "AL" (2)Pass - "Alabama" = "Alabama" (3)Pass - "\$40,960" = "\$40,960" CD(2,x,3) = (1)Pass - "AK" = "AK" (2)Pass - "Alaska" = "Alaska" (3)Pass - "\$54,040" = "\$54,040" CD(3,x,3) = (1)Pass - "AZ" = "AZ" (2)Pass - "Arizona" = "Arizona" (3)Pass - "\$37,960" = "\$37,960" CD(4,x,3) = (1)Pass - "AR" = "AR" (2)Pass - "Arkansas" = "Arkansas" (3)Pass - "\$41,530" = "\$41,530" CD(5,x,3) = (1)Pass - "CA" = "CA" (2)Pass - "California" = "California" (3)Pass - "\$58,240" = "\$58,240"</pre>

Results of the comparisons are shown along with Row Results (RR) and the Total Overall Result (TOR). Providing high level to low level results gives added flexibility in reporting.

- **Analyze the Comparison Results** – With the test results stored in the Complex Data Array, many opportunities are available for further analysis and presentation of the data. Consider the data as being in a type of spreadsheet format that allows for great flexibility.

Test Automation and the challenges of Complex Data

Foundational Principles of the Complex Data Process

The Complex Data process is made up of a number of core foundational principles. These principles have allowed this process to be adapted to a variety of challenges with minimal or no deviations of the Complex Data Process itself.

- **Consistent Referencing of Table Grids** - The Table Index Key (TIK) is a numeric key (example 5,2) derived by executing the WebTableDiscovery function and reviewing the resulting spreadsheet document (see below). The WebTableDiscovery function simply breaks a page down in a consistent manner and provides references to the information contained on the page. The TIK value is comprised of these references, allowing the targeting in on specific Table Grid Objects.

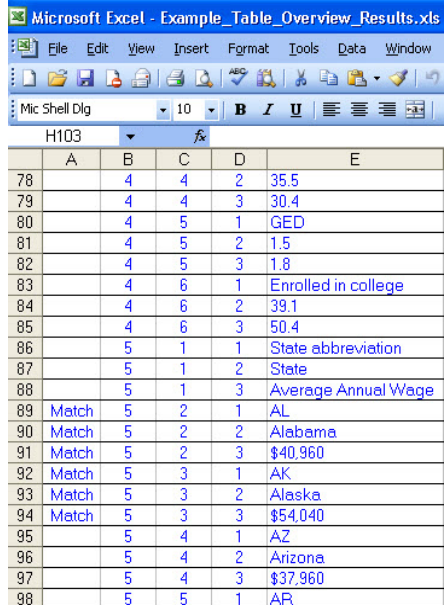
The WebTableDiscovery function provides an easy way to include (pipe delimited) search terms you are searching for within the overall page. When the provided search terms match the information on the page, it's flagged as a "Match" making it easier to find the TIK when doing the post visual analysis of the resulting spreadsheet.

Pipe Delimited Search Terms
<code>."AL Alabama \$40,960 AK Alaska \$54,040"</code>

An example of using the pipe delimited string to help identify specific matches in the Table Grid Object.

The WebTableDiscovery function is generally only needed in the early stages of automation development since once the TIK value is derived, it typically does not change. The exception to this can be if the page design containing the Table Grid Object is modified.

Results of the Web Table Discovery Function		
State abbreviation	State	Average Annual Wage
AL	Alabama	\$40,960
AK	Alaska	\$54,040
AZ	Arizona	\$37,960
AR	Arkansas	\$41,530
CA	California	\$58,240
CO	Colorado	\$45,990
CT	Connecticut	\$59,530
DE	Delaware	\$52,580
FL	Florida	\$49,620
GA	Georgia	\$48,590
HI	Hawaii	\$51,390
ID	Idaho	\$46,330
IL	Illinois	\$59,250
IN	Indiana	\$47,790



The screenshot shows a Microsoft Excel spreadsheet titled "Example_Table_Overview_Results.xls". The spreadsheet has columns A through E. Column A contains the word "Match", column B contains the number "5", column C contains the number "2", and column D contains the state name. Column E contains the average annual wage. The matches are as follows:

Match	5	2	State	Average Annual Wage
Match	5	2	Alabama	\$40,960
Match	5	2	Alaska	\$54,040
Match	5	3	Arizona	\$37,960
Match	5	4	Arkansas	\$41,530
Match	5	6	California	\$58,240
Match	5	3	Colorado	\$45,990
Match	5	6	Connecticut	\$59,530
Match	5	1	Delaware	\$52,580
Match	5	1	Florida	\$49,620
Match	5	2	Georgia	\$48,590
Match	5	2	Hawaii	\$51,390
Match	5	3	Idaho	\$46,330
Match	5	3	Illinois	\$59,250
Match	5	4	Indiana	\$47,790

Above on the left, the Table Grid Object showing details to be analyzed by the WebTableDiscovery function. On the right, the results returned by the WebTableDiscovery function presented in a spreadsheet. Matches can be seen in this spreadsheet (Column A) helping to identify the TIK value, which in this case is 5,2.



It's a good idea to remark out functions used in the initial setup process. This allows them to be utilized for debugging at a later time.

Test Automation and the challenges of Complex Data

A secondary function, WebTableData was created to verify that the TIK value derived from the WebTableDiscovery function is returning the desired results. The WebTableData function generally only needs to be executed in the early stages of automation discovery.

Results of the Web Table Data Function		
State abbreviation	State	Average Annual Wage
AL	Alabama	\$40,960
AK	Alaska	\$54,040
AZ	Arizona	\$37,960
AR	Arkansas	\$41,530
CA	California	\$58,240
CO	Colorado	\$45,990
CT	Connecticut	\$59,530
DE	Delaware	\$52,580
FL	Florida	\$49,620
GA	Georgia	\$48,590
HI	Hawaii	\$51,390
ID	Idaho	\$46,330
IL	Illinois	\$59,250
IN	Indiana	\$47,790

	A	B	C	D	E
1	Table	Row	Col_1	Col_2	Col_3
2	5	2	AL	Alabama	\$40,960
3		3	AK	Alaska	\$54,040
4		4	AZ	Arizona	\$37,960
5		5	AR	Arkansas	\$41,530
6		6	CA	California	\$58,240
7		7	CO	Colorado	\$45,990
8		8	CT	Connecticut	\$59,530
9		9	DE	Delaware	\$52,580
10		10	FL	Florida	\$49,620
11		11	GA	Georgia	\$48,590
12		12	HI	Hawaii	\$51,390
13		13	ID	Idaho	\$46,330
14		14	IL	Illinois	\$59,250
15		15	IN	Indiana	\$47,790

At the left is the same Table Grid Object used in the previous step, this time though we are going to compare it to the results of the WebTableData function. To the right can be seen the spreadsheet containing a perfect match of the data seen in the Table Grid Object providing confirmation that the TIK value (5,2) is indeed correct.

- Consolidate the Data Strategy** - Storing the data and supporting information about the data in one unifying object provides many advantages. This object is represented as a multi-dimensional array, called the Complex Data Array. Each slice of this array has a specific purpose such as storing details about the data, the data itself and the results of comparing the data to the expected results. The Complex Data Array is made up of rows and columns, just like the data on the screen and provides a very logical way of storing and organizing all this information.

One of the biggest advantages of storing everything into one single array is that it can be passed into and returned from function calls. This allows the function to have everything necessary to work with the data in the array including the TIK, Titles, Row and Column counts and Expected Results. Now the function can perform actions upon the data and pass back the data to the calling code for further processing as necessary.

- Functions for Key Tasks** - Creating functions to perform key tasks such as gathering validation data, loading expected results and verification was a valuable “take away” from the initial attempts to develop this process. Because the data strategy was consolidated in the Complex Data Array, these functions could be written and refined, continually improving their performance and robustness over time. This modularization meant that refining just one individual function could greatly impact the overall execution time while adding more reliability.



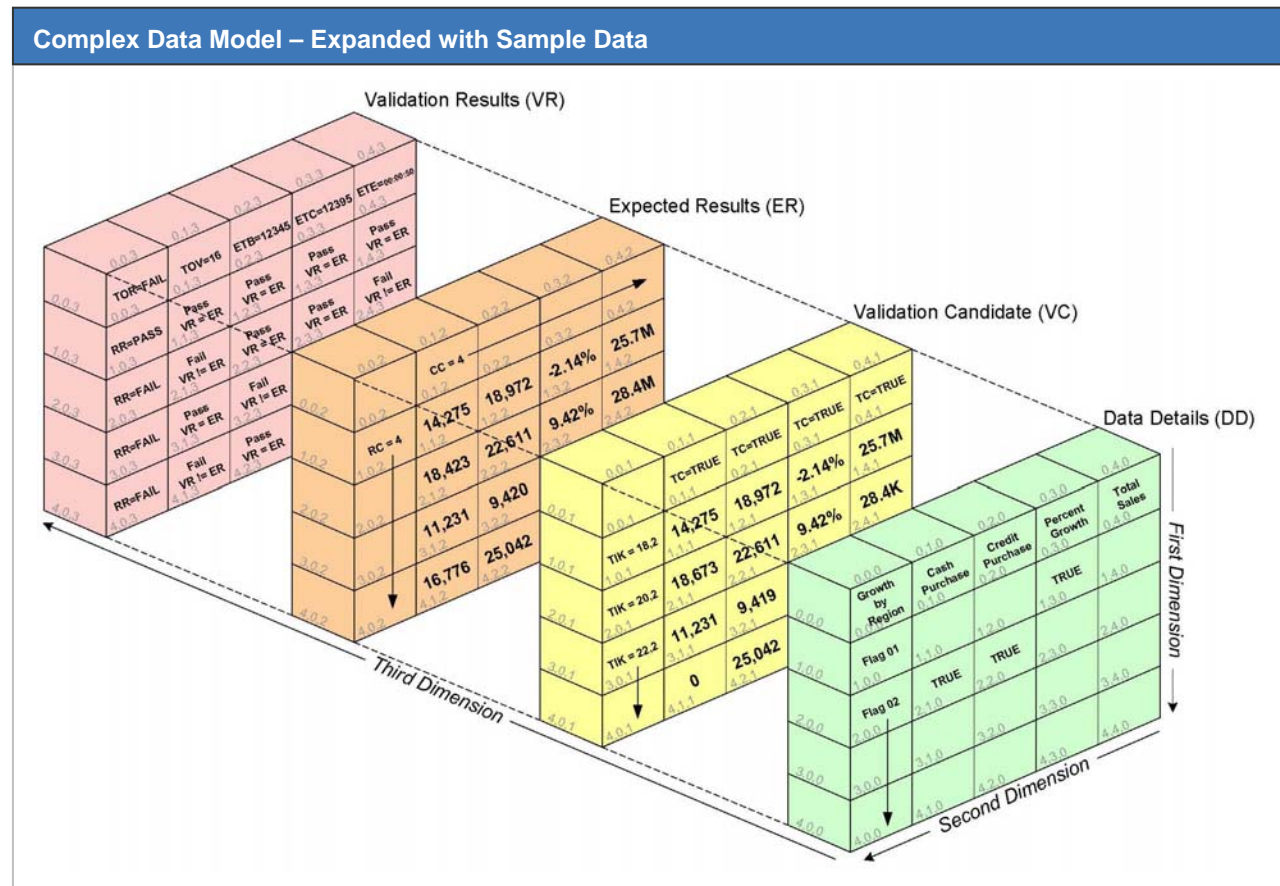
Remember the Complex Data Process is going to be utilized with large volumes of data. Trimming off sub-second amounts of time from a single function can add up to a significant time savings overall.

Test Automation and the challenges of Complex Data

- Consistent Implementation** - Utilizing the above principles, the implementation of the Complex Data Strategy is very easy and consistent. Once the TIK is derived, it is stored along with basic details such as Column Titles and setting basic flags as to which columns of data are test candidates. Functions are in place to gather the Validation Candidate data based upon the TIK provided and validate the results. Most efforts are in determining the appropriate strategy for gathering Expected Results data. In some cases this is data passed in from a data sheet or can be as elaborate as pulling it in from a database.

Visualizing the Complex Data Array

Below is a visual model of the Complex Data Array, it's made up of a number of parts, each with a specific role in fulfilling the overall process. Consider the Complex Data Array as a programmatic representation of this Complex Data Model; they are essentially one in the same, one in concept and one in execution. Let's go into more detail as to what makes up the the Complex Data Model.



A full size version of this diagram can be found in the appendix.

Complex Data Model - Abbreviation Reference

Abbreviation Reference

- **DD = Data Details** – The Details about the data such as Column Titles.
- **ER = Expected Results** – The data you anticipate will be found in the application under test.
- **RC = Row Count** – The total number of row of data.
- **CC = Column Count** - The total number of rows of data.
- **RR = Row Result** - Set to TRUE if all validation results for row are TRUE else FAIL if any validation result for row is FAIL.
- **VR = Validation Results** – The result of comparing the Validation Candidate against the Expected Result.
- **TOR = Total Overall Result** - Set to TRUE if all row result entries are TRUE else FAIL is any row result is FAIL.
- **TOV = Total Overall Validations** – Total number of validations completed. (Excluding TC = FALSE columns)
- **ETB = Execution Time Begin** – The Epoch time the Complex Data process began.
- **ETC = Execution Time Complete** – The Epoch time the Complex Data process completed.
- **ETE = Execution Time Elapsed** – The total time (HH:MM:SS) the Complex Data process was utilized. (Based on ETB and ETC)
- **VC = Validation Candidate** – The data derived from the application under test.
- **TIK = Table Index Key** – A value utilized to locate specific rows of data and derived from the WebTableDiscovery function.
- **TC = Test Candidate** – TRUE = validate column, FALSE = don't validate this column, PREVAL = Column contents have been pre-validated.

Test Automation and the challenges of Complex Data

- **Major Sections** – There are four major sections of the Complex Data Model which are labeled in the diagram, each in a different color. Each section has a primary role but can also assist in fulfilling other parts of the overall Complex Data Process.
 - **Data Details** – Primarily informational in its purpose, this section is used to store details about the data such as columns titles.
 - **Validation Candidate** – Where the actual information pulled from the screen will be stored.
 - **Expected Results** – Whatever method you choose to derive you expected results, this is the section of the Complex Data Model where you will store that information.
 - **Validation Results** – Where the Pass or Fail status is recorded when the Validation Candidate and the Expected Results are compared.
- **Details about the Data** – In addition to simple columns titles stored in the Data Details section, the Expected Results section stores row and column counts. The row and column counts are derived through function calls and can be manipulated within your scripting as necessary.
- **Accessing the Data** – The Table Index Key (TIK) is stored in the Validation Candidate section. In most cases, only one TIK is needed. However, occasionally additional TIK's are necessary to gather all the data that make up a single Table Grid Object.
- **Testing the Data** – Test Candidate (TC) flags can be found in the Validation Candidate section. These can be set to either True or False depending on your testing needs on a column by column basis.
- **Results of Testing** – High level or granular results can be found in the Validation Results section. Total Overall Results (TOR) can be found in this section and provide a bird's eye view of the overall test results. Row Results (RR) as well as individual Validation Results (VR) can be accessed.

Identifying Opportunities

As you approach new Test Automation opportunities, keep the following things in mind to help you find good candidates for leveraging the Complex Data Process:

- **Data Represented in Rows and Columns** - The larger the volume of data represented in a row and column format, the higher likelihood that a good return on investment will be realized utilizing the Complex Data Process.
- **Utilize the WebTableDiscovery Function** - Once you think you have identified a possible candidate, use the WebTableDiscovery function to determine if your testing tools can successfully identify the Table Grid Object.
- **Awareness of Expandable Data** - Leverage the Complex Data Process on Ajax derived data that may require drilling into, to reveal the actual data. It can be more difficult to perform the WebTableDiscovery function upon these types of objects but typically through some creative, descriptive programming, the expanded object and its contents can be accessed. I often refer to this type of hidden data as "Secondary Data".

Test Automation and the challenges of Complex Data

Lessons Learned

As I completed writing this White Paper on the Complex Data Process, I realized I had learned a lot of lessons along the way and I wanted to share a few of them with you.

Continuous Improvement - An overall mindset that has helped more than anything in developing the Complex Data Process is to keep asking the question "What can be improved?" The initial version of the Complex Data Process was clunky and hard to follow but it did prove out the concept. Set the expectations early when exploring a new idea like this that you will refine, once you prove the concept is viable.

Proof of Concept - Building out proof of concept scenarios is a regular way of operating for me and I would encourage you to do the same. Often times we don't know what we don't know and basic experimentation reveals to us the next steps to take. The Complex Data Process is a living process, constantly improving and providing insight into new possibilities with the tools at our disposal. One area in particular that I am certain I will take advantage of in future projects is the idea of utilizing arrays to pass related information into functions.

Modular Design – By keeping the coding design modular, it allows for the reusability we should come to expect from a mature process. Modularity that is not too over specialized yet not over featured is a real art and something that typically comes with time and experience. I find being a visual learner aids in my interpreting coding challenges into logical pieces. Physically drawing these out and looking for similarities helps to identify core functions that should be developed. Refining these core functions with the robustness to deal with anticipated contingencies only adds to their value and adaptability with each coming project.

Economies of Scale – A decade ago, I learned the term **Economies of Scale** and how it applied to companies leveraging large volume purchasing to get a better price which was passed on to their customers. I found this same principal applies in Software Engineering but in this case, it's a savings of time which ultimately translates into financial savings. My early version of the Complex Data Process was developed in a traditional linear fashion, getting to data, gathering data, checking it against expected results and recording those results. This approach got the job done but not necessarily very efficiently. I realized that to gather a single value from a screen contained almost all the effort necessary to gather the next value. I named this paradigm "Like" data meaning to get one value was just like getting another with very minor variations.

When re-engineering the Complex Data Process I decided to take the approach that I would try to perform as many "Like" task as I could in batches and see what the result was. As it turned out, this resulted in less overall code with faster overall execution times especially compared to the traditional linear process. It also resulted in the modular design advantages I spoke of earlier.

Final Thoughts

Look for ways to apply the principles of the Complex Data Process in areas of your coding. Software Engineering still has many unexplored areas where things can be implemented faster and smarter. Consider who might need to maintain things going forward and what it would take to convey the coding strategies you have developed to others.

I hope you have found this White Paper useful and that you will have great success in applying the principles outlined here. Please feel free to reach out to me with questions you have and successes you've achieved utilizing the Complex Data Process. I look forward to your correspondence.

- Greg Paskal

Professional Information

Greg Paskal - JCPenney
Application Test Center - COE
GPaskal@JCPenney.com

Personal Information

Greg Paskal
Greg@GregPaskal.com
www.GregPaskal.com

Test Automation and the challenges of Complex Data

Appendix - Terms

Complex Data Process	An approach to working with large volumes of data represented in Table Grid Objects through test automation.
Complex Data Array	The programmatic object that contains the data and information described in the Complex Data Process. Normally, only one Complex Data Array is needed and it can be reused and resized for other Complex Data Objects. There are occasions when multiple Complex Data Arrays are appropriate such as when it's necessary to interact with more than one Table Grid Object at a time.
Complex Data Model	A visual representation of the Complex Data Array. This model was created early in the development of the Complex Data Process to aid in understanding the data stored in the Complex Data Array and the relationships between that data. Think of the Complex Data Model as a multi-dimensional cube of information, each slice of the cube designated for a specific aspect of the information.
Table Grid Objects	Screen objects where data is represented in rows and columns similar to what might be found in a spreadsheet. Large volumes of data are often displayed in these Table Grid Objects to best organize related information.
Table Index Key (TIK)	The Rosetta stone of the Complex Data Process, the Table Index Key or TIK provides an exact point of reference to where the data of the Table Grid Object begins. With the TIK, you can access and traverse the data in the Table Grid Object. The TIK value is derived utilizing the WebTableDiscovery function developed by Greg Paskal.
Secondary Data	The data that is revealed when a web page object such as a plus sign "+" is selected and additional data is then displayed.

Test Automation and the challenges of Complex Data

Appendix – Complex Data Model

