A Deep Dive Into A Data-Driven World Of Test

Neil Baliga
Verifide Technologies, Inc.
San Jose, CA 95138
neil.baliga@verifide.com

***Abstract*— Traditional methods of testing were based on Instrument-Centric models which included sensor and measurement capabilities inside of instruments that allowed test engineers to be able to perform measurements and collect data. Such data would then be analyzed by software to determine whether the test resulted in a pass or fail disposition.**

Modern technologies have significantly changed the equation into a Data-Centric model whereby data, and lots of it, are the key to gain insight into production and ensure quality. However, most test systems struggle to handle the data that is created by their systems.

This paper will discuss three topics regarding data; a) Data Management – flat files, databases and the pros and cons of each, b) Data Visualization– viewing reports and charts as well as higher level analytics (pareto, cpk), and c) Data Validation – post-test limit checking and yield analysis.

I. INTRODUCTION

We are said to be living in the Information Age today, a period that was marked by the transition from the Industrial revolution to one that is dominated by computers and information systems. Through rapid innovation, computer systems have placed information as a valuable resource that can be compared to the gold rush. Many new tech unicorn companies are building products based on nothing more than information with nothing physical to sell.

However, despite great advances in digital data, the ability to process and make meaningful sense of data is lagging the advances in the ability to collect it. Data is to information as atoms are to matter.

It is all around us in various forms; from physical to digital formats, the concept of data is still the same – it provides a building block for information. Another key aspect of data is its relation to time; data is a small piece of evidence at a given snapshot in time. If not saved somewhere, it has no value in the future.

In prior generations, the reverse was true – ability to analyze data were higher than the means to collect and store data. Test systems were composed of manual or automated test equipment. Data from these systems was acquired through human observation or tabulation and computer systems were used primarily for processing relatively small amounts of data. Large amounts of data were not stored in digital form because of the high cost of disk storage and memory [1].

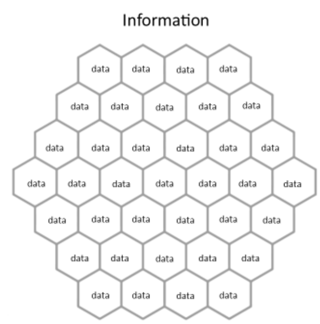


Figure 1- Information is a composition of data

Because of these reasons, test systems were traditionally instrument-centric; meaning they relied on the instrument to acquire, process, and provide a result. Computers were mostly used to schedule and run tests, and higher-level data analysis was not common. Data created during the course of the test was only used to create a disposition and a paper report was generated, because paper was much cheaper than disk storage.

In heavy industries like aerospace and defense, harnessing data for more efficient manufacturing processes was not a priority. With the global market today, these industries are under new pressures in terms of cost, schedule, and competition from countries with lower manufacturing costs who are advancing their technical capabilities in these sectors [7].

A lot of test systems in these sectors are also aged, so they were developed at a time when capabilities to store and manage large amounts of data did not exist.

Things have changed significantly in the past 5-10 years in terms of the ability to store and manage digital data. This means that test systems can now store all the data they want and many of them store more than they need. This is primarily because of the temporal nature of data – it is better and cheaper to collect data now and then figure out how to use it later.

Yet, despite all the big-data buzzwords thrown around Silicon Valley, the ability to adapt any of these advances to test data has been difficult. The paper will cover three key aspects of test data and some solutions that will help test systems catch up to the digital data revolution- a) Test Data Management, b) Test Data Visualization, and c) Test Data Validation.

II. DATA MANAGEMENT

The overall process for testing involves acquiring data, processing it, analyzing it, then presenting it such that a decision (manual or automated) can be made.

In that process of testing, there is various amounts of data that go through stages. In its purest form, data is a physical quantity. It then becomes digitized by a sensor and then gets transformed through test equipment, test code, and processing algorithms. The end result is another piece of data that is used to make a decision.

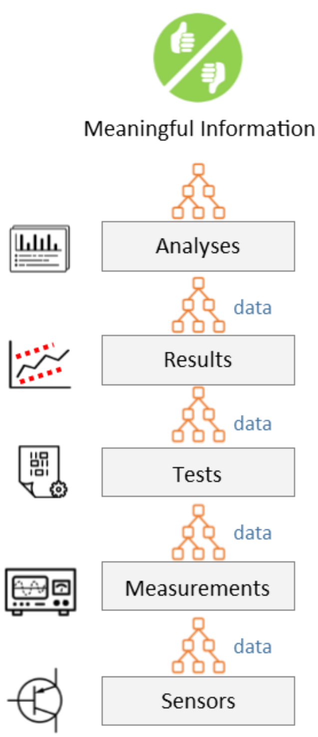


Figure 2 - Data is transformed up the test value-chain

The nature of testing is that data evaluation methods change often, new algorithms are constantly being developed and there is need to be able to iterate. Also, if there are anomalies, the evaluation and diagnostic logic may be completely different and would require non-standard processing to get at the root cause of the anomaly.

For this reason, data that is taken during testing is of extremely high value as it can tell many stories about the system being evaluated. Test systems need to store raw data from measurement equipment, calibration data that is used to offset measured results, test settings, limits, input parameters, and of course, the final processed data in digital form as well as reports.

The importance of retaining data on aerospace and defense systems and components is even higher [3]. This is because of a few factors – a) the systems are more complex and often require significant analysis and troubleshooting at times b) the systems are high-reliable devices, so the evaluation needs to be exhaustive, and c) the life-span of the devices is longer than other industries, which means that the data will be the key in analysis that will be performed much later into the future.

A. Why Is Test Data So Hard To Store and Manage?

Since the 80's there have been numerous advances in data engineering and in managing "big-data" and analytics. Banking and financial systems, web technologies, cloud storage and others have, relatively successfully, managed to implement very sophisticated systems that are dealing with large amounts of data. This has been done using various Relational Database Management Systems (RDBMS) and the capabilities over the years have been scaled mostly through innovations in this field [4].

The problem with TEST data is that there is no unified model that can be applied to data like banking or eCommerce. A factory can have numerous types of testers. Each tester type can be composed of 10's or 100's of individual test types. Each test, within each tester, has its own data that it collects and stores.

To store test results data, there are mainly two options that developers have: use an RDBMS, or use flat files.

1) RDBMS Systems

In the software world, relational databases are the typical means of managing large amounts of data. To put test data into a RDBMS database, it must be a) designed with schema, b) created through SQL code, and c) populated with data by the test code using SQL instructions.

The benefit of getting data into the database is that it provides a great amount of analytical capabilities with queries and data processing that would otherwise be difficult to do with large amounts of scattered individual files on a disk.

The main difficulty with databases are that they are stubbornly rigid. Tests are routinely updated and new tests are constantly being added to systems. If the test needs to store additional data, or needs to change the types of data it stores, then it requires a monumental task of re-designing the database schema and updating code to create and store data in the new schema. Furthermore, databases do not offer versioning, so an organization would need as many database implementations as they have schema versions, not to mention multiple concurrent versions of software that target different schemas.

For these reasons, databases have been largely prohibitive for most test organizations because of the vast overhead in upkeep and iteration (and lack of DB expertise in-house). Test organizations that implement databases are trading a lack of flexibility for far better analytical capabilities.

2) Flat Files

Because test data is so hard to implement with databases, most organizations defer to the alternative and store their data in flat files. A flat file is basically a single file that stores data from one or more tests onto a disk. The file format itself varies considerably across industry, but can range from custom binary, plain ASCII text like CSV, XML, or other variants like ATML, and HDF5 [5].

Flat files offer the best flexibility with the data because it does not require huge changes to databases or specialized database engineering. Developers can also get started writing tests quicker and data files can mostly be viewed and edited using standard desktop tools (Notepad, Excel, XML viewers). The difficulty with flat files comes not during testing, but rather after testing is done because using flat files means trading away analytical capabilities for better flexibility with storing data.

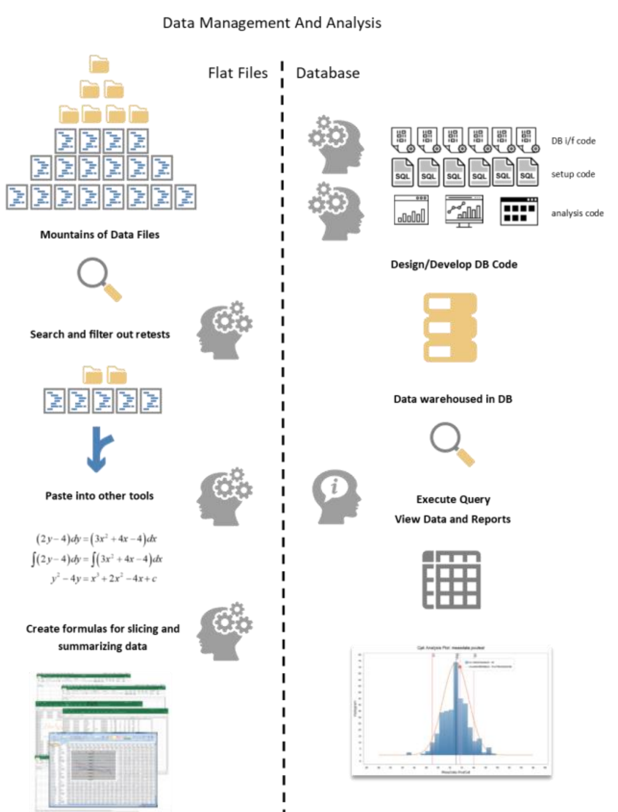


Figure 3 - flat files and databases both have pain points

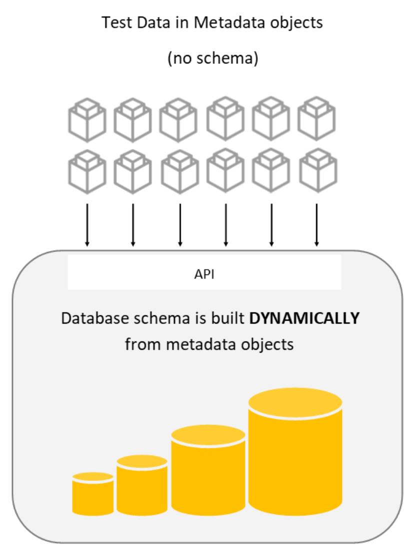
3) Solving the Problem

To begin to solve the problem, we must first recognize root of the problem; **schema dependency.**

Considerable work has been done at the standardization committee level, to attempt to develop common flat file formats and structures (ATML, HDF5). However, these formats carry the same fundamental problem – locking test systems into a schema that requires significant changes to handle iteration and growth. Can a schema that was designed for board testing really work for on-wafer RF measurements? Can it survive another 5-10 years of maintenance of the system?

With databases, schemas are also very problematic because the cost of implementation is significantly higher than the effort to adopt a new XML schema – it requires DB skill sets, IT administration, backups and recovery compatibility, and modifications to custom analysis software. The costs associated with multiple versions of schemas are considerable, given that organizations typically need to support side-by-side versions of tester families.

Verifide has achieved a breakthrough in this area by implementing a **schema-free** database structure that allows developers the flexibility of flat files, but with the analytical capabilities of a database.



The solution is based on three key concepts:

a) Test data is represented in a flexible markup type metadata structure requiring **no schema**. Developer chooses what data to store – arrays, images, numeric values etc.

b) The software layer above the commercial RDBMS translates the metadata and stores in the DB. No schema or database setup is needed – it is dynamically adapted to the metadata. Updates to database indices and tables are done automatically by the software.

c) The analytical capabilities are developed on a flexible metadata structure- allowing trending, reporting, and statistical analysis to be done without custom code.

III. Data Visualization

Data is the foundation of being able to make sound decisions [6]. However, it needs to be transformed into meaningful information to make *better* decisions.

In previous generation test systems, a test report was generated showing the output of a test or group of tests. The report was created by custom software that created report pages and charts that showed the data visually for human analysis. These reports were often printed on 3-hole punched paper and inserted into sell-off packages and kept as the official test record.

Modern manufacturing needs to be much more efficient at analyzing data; factories need to be able to prevent failures in the field and fine-tune processes for longer term maintenance of systems. This efficiency can only be achieved through higher level analysis of the data that can slice, aggregate, and visualize data across all the individual test results.

There are three types of data visualization that enable analysis at the higher levels.

A. Trending

Individual tests typically store multiple data fields that were used to disposition the test. These could be raw values, calibration values, and results that were determined after applying analytical algorithms in the test code. Tests are limited in their evaluation scope because they only have the responsibility of evaluating the data for that test.

To get a better picture of the behavior of a product, it is necessary to see the performance of test data across multiple tests. For example, a production run of 1000 devices results in a yield of 99.99%. However, upon further inspection, it shows that the latter 50% of units tested are not stable and the data is bouncing around very close to the limits. Though all the data passes limits, it is not a decisive indicator that there are no anomalies, or that failures are less likely.

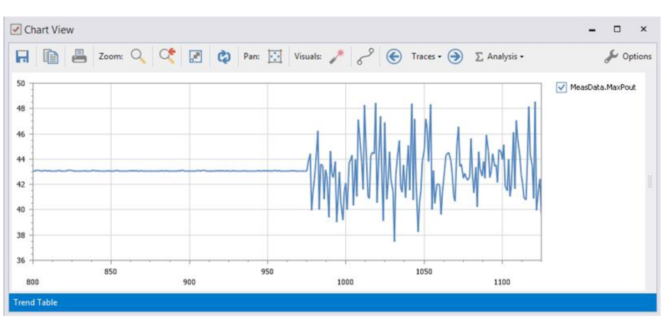


Figure 4 - Visible indication of an unstable system with trend charts

To be able to assess the production batch, an engineer would have to see a visualized graph of all the data values plotted against a time line or across other variables like test station, or serial etc. This visualization will provide the basis upon which to examine whether the anomaly is due to a different vendor part, manufacturing process change, or test system error, for example.

B. Statistical Processing Control Reports

Statistical processing control (SPC) analysis is a different type of analysis in that it specifically applies statistical models and analysis to the data. This mathematical field of statistics is all encompassing and has applications in almost everything around us from electronics, to health science, insurance, finance, and politics.

With regards to testing, statistical processing has traditionally been applied to large volume manufacturers such as semiconductor [2]. These manufacturers use these reports to know the variation in measured data across the production lot. A Cp and Cpk analysis, for example, is a measure of the process capability and how "normal" the data behaves to a statistical bell curve.

Other plots like tolerance interval plots allow a manufacturer to determine, for example, the range of measured values where X% of data is likely to fall between with a Y% statistical confidence.

The basis of statistical theory follows that the accuracy of the model is improved through higher sample rates. High-volume manufacturers can use these reports because they deal in higher volumes and therefore can validate their models.

That doesn't mean that statistical processing is not useful in other types of testing such as aerospace and defense because these industries do employ high test volumes even if they don't necessarily produce higher volumes of devices. These industries are typically high-mix, meaning that the test count per device is orders of magnitude larger than the number of devices produced. Think a single aircraft and the amounts of test data at the component, sub-assembly, and system level.

Another key use case for statistical processing is not for manufacturing itself, but rather for the long-term maintenance of the product. For organizations like military depots that maintain devices for extended periods of time, statistical

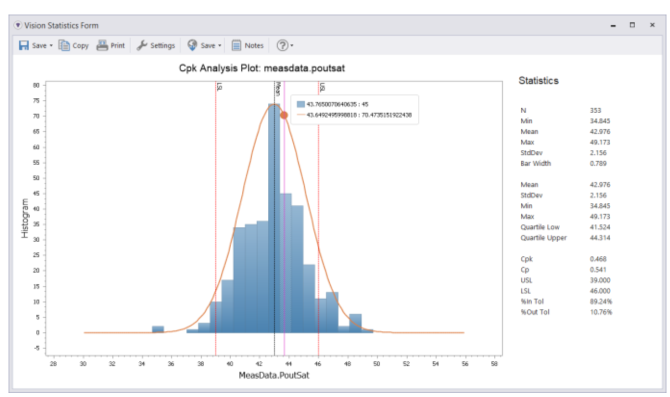


Figure 5 - Statistics have many uses beyond process control

insights into the trends and variance of data across devices allow them to adjust limits and tolerances for equipment repair and service, as well as detect field problems with units earlier.

Though predictive modeling may be less accurate at lower ***device*** volume, there is still significant benefit to SPC so long as the ***data*** volume is high.

a) Process control – SPC is still useful for displaying the distribution of data that will indicate whether the data is well behaved or not

b) Troubleshooting – When troubleshooting, live access to data distributions will help to place whether a data excursion is a common event or if it is out of the norm.

c) Limits Tuning – By seeing the spread of data and assessing the normality of that data, test organizations can fine tune limits around the mean and account for biased distributions.

d) Maintenance – Seeing the spread of data allows maintenance organizations to better support units and to anticipate and prevent problems in the field.

C. Metrics

Tests results contain lots of engineering data such as voltages, powers, bits etc. However, each test carries other business-related data that is also valuable for improving efficiency.

For example, a production lot of 1000 devices are tested using 20 test stations. Each device has multiple tests ran on it, and the entire production takes about 6 months. To assess the performance, the test engineering manager wants to see, for each device serial number, the total number of tests and the total test time in the last month of production. The manager also needs to see the test counts and test times for each test type ran on the device.

This information will give the manager the information he or she needs to improve the production throughput and reduce cost and schedule. For example, with metrics, the manager can see that systems are not being utilized at capacity, so adding new capital test equipment will not speed up production. The manager can also see that the number of retests is too high, so improving some test code would likely reduce production time.

When faced with large volumes of data that have multiple attributes, the analysis is not simple. It is not sufficient to see that all data in a gigantic spreadsheet with rows and columns beyond the user's visual range. To analyze that data, it must first be sliced in targeted ways before summaries can be calculated to give the user meaningful information.

The best way to obtain these metrics, is to visualize summaries of multi-dimensional data is using Pivot Tables. Pivot tables are a cross-tabulation of data that results in summaries for each variable field in the analysis. The table shows summaries such as count, sum, average, standard deviation computed across data that is sliced into multi-level

Count(Run) / Sum(#Mins)		Status: Fail		Pass			
Serial	Test Name	Count(Run)	Sum(#Mins)	Count(Run)	Sum(#Mins)	Count(Run)	Sum(#Mins)
UNIT_1	DemoTestGain	208	190.7100	1064	917.5300	1272	1108.2400
	DemoTestPower			51	0.0000	51	0.0000
	EmissionsTest			47	0.1800	47	0.1800
	GainTest	42	38.2300	666	510.3000	708	548.5300
	PowerTest	24	19.3700	578	460.8700	602	480.2400
	StepAccuracyTest	26	22.3000	529	484.0100	555	506.3100
	StepSpeedTest	35	31.5500	591	492.4600	626	524.0100
UNIT_1 Total		398	347.7900	4092	3355.8800	4490	3703.6700
UNIT_2		402	354.0700	3984	3329.7800	4386	3683.8500
UNIT_3		399	330.8400	3950	3323.3200	4349	3654.1600
Grand Total		1227	1032.7000	12313	10008.9900	13540	11041.6900

Figure 6 - Pivot tables contain summaries by custom groups

groups.

Though the pivot table provides summary information, it is better presented in a visual form using histogram and pie charts. The histogram charts provide a visual indicator that can delineate the variables in the analysis and present the differences and dependencies with more emphasis.

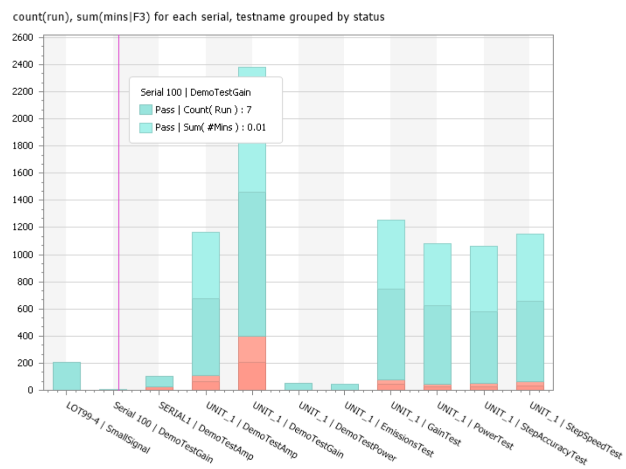


Figure 7 - Pivot charts show grouped data in a visual perspective

Some common examples of how metrics from test data can be useful in a business case are:

a) Forecasting – this will allow managers to see test volume breakdowns to determine if projects are on-schedule or if not, the manager can take preemptive steps to correct.

b) Resource Management – using pivot tables, a manager can identify underperforming tests or tests with high retest counts, and prioritize resources to fix accordingly.

c) Bidding – using data metrics, the manager can determine actuals from past projects and predict future production runs more accurately.

g) Process Improvement – with metrics, managers can identify performance bottlenecks such as low utilization of testers, process variation across testers, and even view correlations to sub-assemblies and components.

D. Challenges with Visualization

The data visualization forms mentioned above are a means to reach the end – to make a good decision about the products being tested. Organizations that need this information develop in-house solutions to extract the data from vast troves of flat files, and either use some commercial tools like Excel for plotting, or build their own code to crunch these results and create summaries.

The result is that engineers cannot easily analyze data in different ways. They must be pre-defined and implemented and iterating is not possible without additional software or spreadsheet development effort.

Though inefficient, organizations have gotten by with manually labor-intensive analysis. However, new challenges with cost, schedule, device complexity and larger data volumes mean that manual processes are either not manageable anymore, or in the worst of cases they are highly error prone leading to failures that should have been caught.

The ideal solution to this problem would be have a generic data analysis application that did not depend upon the test data schema. It would be able to extract and analyze data that is stored without needing to customize software if the engineer wants to analyze data in different ways.

There are three key things that need to be implemented to enable a generic data analysis application:

a) A generic means of defining what to retrieve and process from the data set.
b) A generic means of extracting only selected data values from the database into an intermediate result set for analysis.
c) A generic set of data analysis applications that operate on the intermediate result set.

Verifide's data analysis software extends upon the schema-free architecture and implements such a generic solution.

A query object can be created by the user that defines the data analysis the engineer wants to perform. The software then creates the lower level database SQL query dynamically and extracts the results into an intermediate result set that can be processed for statistics, metrics, or trending.

The data analysis applications are built to operate on the intermediate results set, so it is independent of the query fields or any schema thereby allowing the applications to work across data of all shapes and sizes.

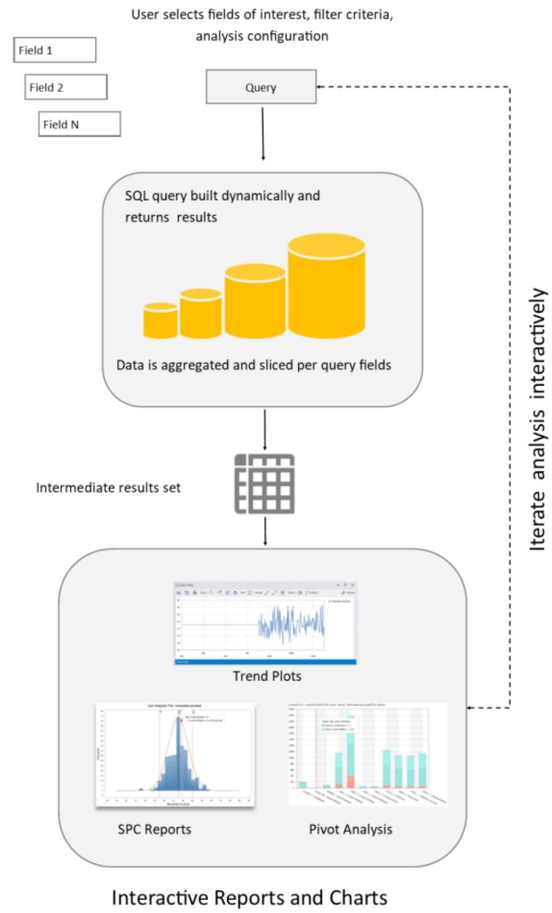


Figure 8 - A way to implement generic data analysis applications that are schema-independent

IV. Data Validation

The very purpose of testing is to evaluate some measured data against defined criteria. This evaluation provides for deterministic analysis of test data. In the days before test automation, an engineer would take readings down from an instrument and tabulate and plot it manually. Then they would look at results against limits and evaluate.

With test automation, we've now made advances where the limit checking of the data for a test is embedded into the test sequence itself. This allows for real-time fault detection and to move production at a faster pace because a disposition for the test result can be done much faster.

The problems with automated limit checking are that limits themselves are fluid in many production environments. For example, a limit set that is a plus or minus tolerance around a predict value may be based on a theoretical mean of the results. There are bias or offsets in production data due to compounding factors and these limits need to be changed.

Limits are also difficult to nail down exactly. If you set them too tight, then you end up with many false positives that need to be manually dispositioned by an engineer. If you set them too loose, you risk failures in the field. Most specification engineers find a balance of limits that they believe are acceptable tolerances.

Some test systems forego the limit checking altogether and just collect data. It is then left up to the systems engineer to post process the results. This often means that anomalous test results are found long after the test run, so devices or tests that need to be rescheduled take up more production time through re-tests.

Another challenge with nailing down limits is that product design and manufacturing variables are constantly changing. Next generation of products may have minor technical iterations, but the impact to manufacturing is significant because test limits and process variation need to be re-established. Test organizations may also upgrade test equipment that has different sensitivity, or products could integrate components from new vendors that introduce other variations that need to be accounted for.

This creates a difficult scenario for data validation because the goal posts are frequently moving.

One solution to this challenge with limits is to institute a generic framework for limit checking that can be ran on data – during testing, or after testing. The ability to perform limit checking after the fact has numerous benefits; production limits can be set looser till limits are mature, engineers can fine-tune limits for desired yield without re-running tests, and maintenance organizations can more accurately plan for repairs or recall devices from the field based on prior data.

Verifide's software implements such a framework by building upon the dynamic database architecture to enable a generic method of limit checking any data in the database. A limits definition file can be setup to process the meta-data that comes from the database. Limit checking can be performed against any sets of data and the engineer can perform hypotheticals with various limit definitions to see the yield.

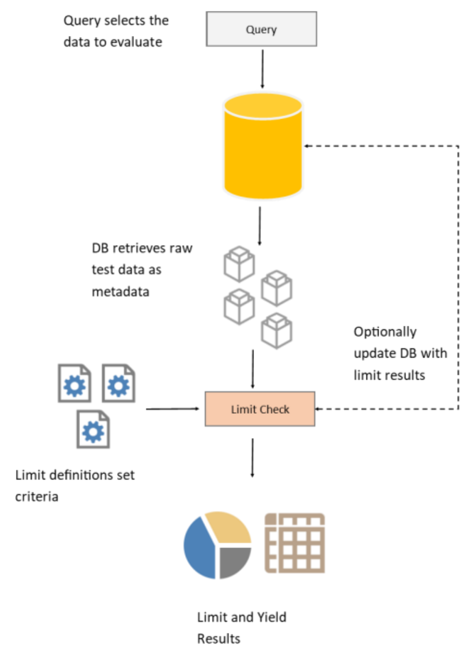


Figure 9 - Ability to post process limits without custom coding

CONCLUSION

This paper provided a deep dive into the world of test data management, visualization, and validation. Advances in other fields with digital data have not reached the world of test because of the problem with schema dependence. This paper presented a break-through solution that we have adopted at Verifide that allows for a schema-free structure where data can be managed in a relational database without any custom implementations and costly iterations.

ABOUT VERIFIDE

Verifide Technologies, Inc. is a software company based in Silicon Valley, CA and founded in 2006. We provide next generation products for test automation and test data analysis. We have developed breakthrough technologies that makes it possible for you to have truly modular and scalable test systems.

Our software is being used in mission critical applications across organizations that have few users to some with hundreds of users. Our software has been used for the past 10 years to test products in various industries including Defense, Satellite, Semiconductor, and Wireless

REFERENCES

[1] M. Komorowski, "A History Of Storage Cost," 2014. [Online]. Available: http://www.mkomo.com/cost-per-gigabyte.

[2] I. Ubaid, "Process Control in High Volume Semiconductor Manufacturing," [Online]. Available: http://anysilicon.com/process-control-high-volume-semiconductor-manufacturing/.

[3] D. Briggs, "Aerospace Industry Requirements For A Long Term Data Retention Process," [Online]. Available: http://proceedings.ndia.org/3AF6/Briggs_AEFI_Presentation.pdf.

[4] "The Future of Enterprise Data: RDBMS Will Be There," [Online]. Available: https://www.wired.com/insights/2013/09/the-future-of-enterprise-data-rdbms-will-be-there/.

[5] "ATML," [Online]. Available: https://en.wikipedia.org/wiki/ATML.

[6] D. B. a. D. Court, "Three keys to building a data-driven strategy," 2013. [Online]. Available: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/three-keys-to-building-a-data-driven-strategy.

[7] R. Lineberger, "2017 US aerospace and defense industry export and labor market study," Deloitte, 2017. [Online]. Available: https://www2.deloitte.com/us/en/pages/manufacturing/articles/us-aerospace-defense-export-competitiveness.html.