Learning the Process for Correlation Analysis

Sebastian Siatkowski, Li-C. Wang
UC Santa Barbara

Nik Sumikawa, LeRoy Winemberg
NXP Semiconductors

***Abstract*—An analytics process is subjective to the perspective of the analyst. This paper presents a learning approach that models the process of how an analyst conducts analytics. The approach is applied in the context of correlation analysis for production yield optimization. The benefit is demonstrated by showing that learning from resolving a yield issue for one automotive product line can help resolve a yield issue for another automotive product line.**

I. Introduction

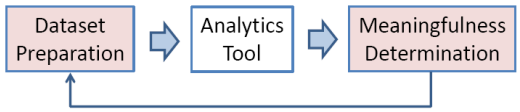

Fig. 1. Analytics can be viewed as an iterative search process

Analytics can be viewed as an iterative search process that comprises three steps: (1) dataset preparation, (2) running an analytics tool, and (3) meaningfulness determination. Analytics has found many applications in test and shown great promises. One example is yield optimization [1] where analytics helped improve the production yield.

In an analytics process, dataset preparation and meaningfulness determination are largely empirical. From the raw data, an analyst decides how to prepare the dataset for the tool to analyze. After the tool outputs a result, the analyst examines the result to determine if it is meaningful.

For example, in the context of production yield optimization [1], an analyst desires to find a high correlation between a process parameter measurement (E-test) and a type of fails. Suppose for an E-test, its *average* measured values over multiple sites on a wafer, across n wafers, are $\vec{e} = \{e_1, \ldots, e_n\}$. On the other hand, the numbers of fails, based on a test bin, across the wafers are $\vec{f} = \{f_1, \ldots, f_n\}$. The analyst prepares a data file of two vectors $(\vec{e}, \vec{f})$. Then, with p E-tests and k test bins, the analyst prepares a dataset of $p \times k$ data files.

Each data file is fed into an analytics tool. For example, the analyst can use a statistical correlation tool from the Scikit-Learn Python library [2] to analyze each data file. The result is $p \times k$ correlation values. The analyst examines these values to determine if any are meaningful.

If none of the correlation values are perceived meaningful, the analyst might decide to construct a different dataset. For example, instead of using each test bin as the basis for a data file, the basis can be each individual test.

In [1], ideas were presented for how to construct a dataset to search for high correlation between E-test and fails. Each construction represents one particular *perspective* for how the correlation might exist. In view of Fig. 1, it is obvious that the effectiveness of the analytics depends on the set of perspectives the analyst has in mind. If the desired high correlation existed only in a perspective that the analyst never thought of, then the analytics process would not find it.

Even though the work in [1] demonstrates that analytics helps improve the yield significantly, the above observation leads to two fundamental questions when the analytics methodology is to be used for another product line:

- What if for another product line, the high correlation only exists in a dataset that requires a perspective we never thought of when conducting the work in [1]?
- What if the task of improving the yield for another product line is given to an engineer with little analytics experience who, for example, has no idea of the different perspectives proposed in [1]?

These two questions motivate the work presented in this paper. Because of these two questions, it is desirable that different ways to prepare a dataset can be learned and generalized by a learning software. If this can be accomplished, the software can become a "surrogate" for the analyst in future tasks after learning how an analyst performs an analytics task. Therefore, in this paper, we present a learning approach to achieve that purpose.

In particular, we will show that the learning approach can learn from the analytics process for resolving the yield issue as presented in [1] and apply the learning model to resolve another yield issue for another product line.

The rest of the paper is organized as below. Section II provides a brief review of the work in [1]. Section III discusses the learning problem studied in this work. Section IV presents the approach for learning an analytics process. Section V explains the software design needed to bring process learning into the context of correlation analysis. Section VI demonstrates the effectiveness of the approach. Section VII concludes.

II. Perspectives in Yield Optimization

A *perspective* is a particular way to construct a dataset. In [1], yield optimization refers to the task of finding a high correlation between an E-test and a set of failing dies. The analysis is wafer-based, meaning that two values are calculated for each wafer, one for E-test and the other for failing dies. Then, correlation is analyzed across a set of wafers. Using two values is for the case when one desires to use a standard correlation tool. If a canonical correlation tool [3] is used, two vectors of values are extracted for each wafer.

For simplicity, take standard correlation as the example. On the E-test side, one can have different ways to calculate the value. For example, a process parameter is measured on multiple sites, say 5. The value can be calculated based on taking the average of all 5 sites or a selected subset of sites.

This work is supported in part by National Science Foundation Grant No. 1618118

978-1-5090-4482-5/17/$31.00 ©2017 IEEE

On the side of failing dies, the choices can be many. For example, the value f_i for a wafer W_i can be:

- f_i is the number of fails due to a test bin (as mentioned before).
- f_i is the number of fails due to a test.
- f_i is the number of fails based on a particular test value.
- f_i is the number of fails based on a test value range.
- f_i is the mean value of a distribution of test values based on a test.
- f_i is the variance value of a distribution of test values based on a test.

In addition to the above diverse choices for the two values, a dataset can be constructed by taking two additional aspects into account: (1) a *spatial* aspect that restricts the population to a selected wafer region, (2) a *temporal* aspect that restricts the population to a set of selected wafers.

A. What contributed to the success in [1]

The work in [1] was for resolving a yield issue for an automotive product line (a sensor product). Prior to the work, attempts for yield improvement were made through one design revision, multiple test revisions, and analytics to find high correlations, but all those attempts failed to improve the yield.

Note that the work [1] presents other important methods (e.g. risk evaluation) than the diverse perspectives for dataset construction, which all contributed to the final success. However a fundamental reason why the earlier analytics attempts failed but the work succeeded was indeed due to the fact that earlier attempts never analyzed the data from the perspectives the work employed.

For example, the highest correlation values found by the earlier analytics attempts were all below 0.5 which were not strong enough for the foundry to change their process.

On the other hand, Fig. 2 shows two example results with (absolute) correlation values both above 0.75 [1].

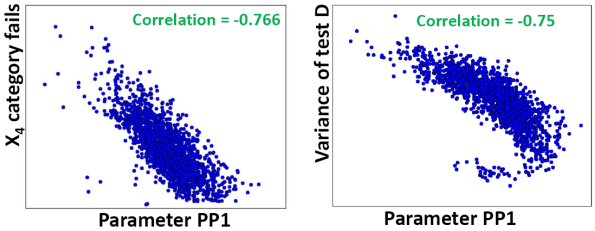

Fig. 2. Examples of high correlations found

In the left plot (every dot is a wafer), the number of X_4 category of fails is correlated to the average E-test value from a process parameter PP1. Note that X_4 denotes a particular test value of a discrete test. In the right plot, the variance of measured values on another test D is also correlated to PP1.

Fig. 3 shows another example result by considering the *temporal* aspect. In this example, parameter PP5 is correlated to the number of X_1-X_3 categories of fails. The wafers are separated into two groups, colored as green and blue dots. The separation is based on the time of their production. Not all wafers are included. The separation improves the correlation values individually from their combined analysis result of 0.63.

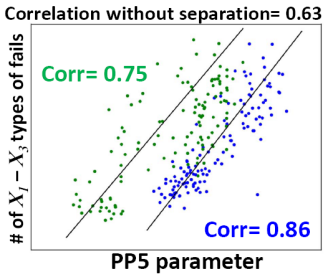

Fig. 3. An example of uncovering the temporal effect

As explained in [1], the high correlation results discovered were later translated into two process adjustments which resulted in significant yield improvement on silicon. The adjustments were therefore adopted for mass production later.

III. The Learning Problem

As pointed out in the previous section, the main reason why the earlier analytics attempts did not discover the results as shown in Fig. 2 and Fig. 3, is that the analysts conducting those attempts never constructed the particular datasets to look for those particular ways to correlate E-test and fails.

Suppose one desires to make a technology deployment of the work in [1] to all the product teams. The common practice today is to implement every perspective employed in the work into a software tool and deploy the tool. However, this approach is not effective.

For a future task, it is possible that the required perspective to analyze the data is not in the set of the perspectives implemented. When that happens, the tool will fail. Then, because the product team usually does not understand the implementation of the software, they will ask the tool developer (the analytics expert) to debug and enhance the tool. This "centralized" approach puts all the burden on the expert.

Instead of providing a fixed set of perspectives, it would be more desirable for the tool to provide a set of toolboxes to enable the product team to conduct the search based on their own perspectives. More importantly, the tool can *record and generalize* from those perspectives and share that experience with other product teams. In this way, the burden is "distributed."

In order to develop such a tool, one needs to answer the key question: "How to learn (record and generalize) from someone's perspectives?"

A. Unsuccessful analytics trials

Learning the perspectives provides another advantage. It is common in practice that an analyst remembers the analytics process instances that lead to good results but forgets those that do not. With the proposed learning tool, all process instances (and their perspectives) can be recorded.

For example, for resolving the yield issue in [1], not only were different types of statistical correlation tried, but different attempts to establish an *association* relationship were also tried. Those association attempts were never reported in [1] because they did not lead to successful results.

1) Unsuccessful example 1: For example, Fig. 4 shows a heatmap association that led to unsuccessful search. The left plot shows a heatmap constructed based on one lot of wafers. The color indicates the number of fails from the test bin that has the largest number of fails. Red means more fails.

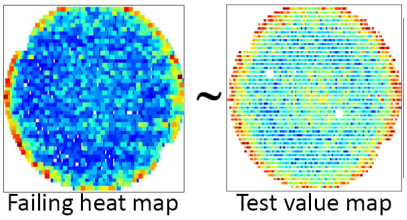

Fig. 4. One example triggering unsuccessful search

The right plot shows the measured values of a frequency test. The interesting point to observe is that the wafer pattern exposed by the frequency test is similar to the failing heatmap. Fig. 4 led us to believe that future search should be based on the frequency test. However, this was not successful.

2) Unsuccessful example 2: Fig. 5 shows another unsuccessful example. The plot shows a 2-dimensional space with two E-tests, P_x and P_y. Each dot is a wafer, positioned by its average measured E-test values.

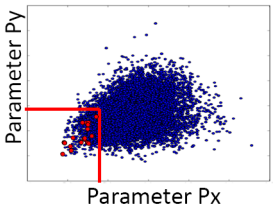

Fig. 5. Another example triggering unsuccessful search

The red dots are the 20 wafers with the lowest yield. Notice that they concentrate on the bottom left corner of the plot. This plot associates lowest-yield wafers to the two process parameters. Hence, it was thought that future search should focus on these two parameters. However, this was wrong again.

Even though the different types of association analysis led to no successful result, it does not mean that they will not be useful for another task in the future. Hence, learning should also take such unsuccessful trials into account.

IV. Learning the Perspectives

As mentioned above, *learning* in this work is to *record* and to *generalize*. The main subject of the learning is the set of *perspectives*, or *ways to construct a dataset*.

In order to learn perspectives, we first need to have a way to represent a perspective. The learning algorithm and the effectiveness of learning depend on this representation.

The representation employed in this work is to view each perspective as a sequence of *steps* to manipulate the data. If each perspective can be represented as a sequence of steps, or a *path*, then Process Mining (PM) [4][5] can be applied.

Process mining was originally motivated by the need to analyze logs from Workflow Management systems in business applications. It has its root in the early research of inductive inference [6] that studies the classes of *learnable* formal languages. For example, in theory, the PM problem is similar to learning a finite state machine (regular language) from the machine's inputs and outputs.

Suppose there are five possible steps denoted by A-E, and there are two example paths, ACB and DCE, that have been executed. The set {ACB, BCE} can be called a *log* file. Each instance in the log can be called a *trace*.

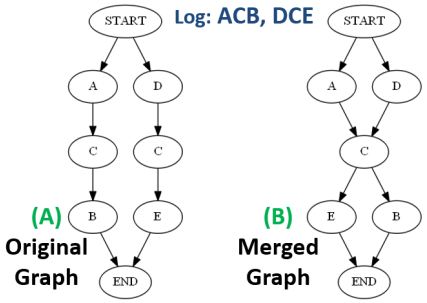

Fig. 6. Simple state merging example

Fig. 6 shows two possible process models that can be learned from the log. Model (A) basically records the two traces. Model (B), on the other hand, recognizes that step C is common in both traces. By merging step C into a single node, model (B) represents four traces, ACB, DCE, ACB and DCE where the latter two are new. In this case, model (B) *generalizes* the log to include new traces.

When the log file becomes rather large, the number of possible merges grows quickly. This phenomena is illustrated using another simple example in Fig. 7.

For example, model (A) like before simply records all traces. Model (B) merges every step with the same name into a single node. Model (B) generalizes from 5 traces to have 27 traces in the model.

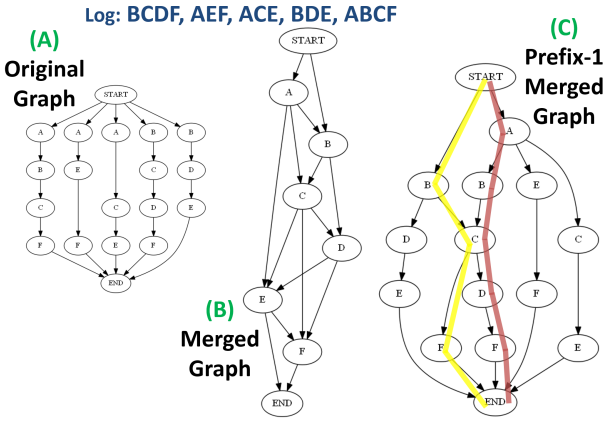

Fig. 7. Another state merging example

Model (C) is based on a *prefix* rule. Suppose two partial traces are pX and qX where p and q are two sequences of steps each containing at least 1 step. Given a length l, let p_l be the last l steps in p and q_l be the last l steps in q. An l-prefix rule means that the two X nodes are merged only if $p_l = q_l$. Model (C) is obtained based on 1-prefix rule.

Notice in model (C) that every sequence is assumed to begin with a special step START and end at a special step END. Therefore, the three A steps in AEF, ACE, and ABCF, are merged into one node. However, this merging does not create

any new traces. The two C steps in BCDF and ABCF are merged into one node because the steps before both C steps are B. This merging creates two new traces in the model, BCF and ABCDF, which are the only new traces in the model.

Using a prefix rule and by controlling the length l, one can control how many new traces are added into the model, i.e. how generalized the model is. This point will be shown later with an experimental result.

Implementation of a prefix rule poses a subtle issue – if two steps are merged, then the prefix for a step that follows the merged steps can be altered. For example, in Fig. 7 model (B), after the C steps are merged, the 2-prefix of step E is altered. In the log, the 2-prefix of step E is only BC. After the merge, the 2-prefix of step E is {BC, AC}. Hence, we need to decide how a prefix rule applies if the prefix becomes a set.

In our implementation, if any pair of prefixes from two prefix sets are the same, we consider the prefix sets compatible and trigger merging the two steps. Experimentally, this enables more (but not too many more) merges to take place, resulting in a more useful model.

The ordering of the merges is based on a partial ordering graph built based on the log. For two steps, X and Y, we define X $>$ Y if Y only appears after X in the log. The merging begins with the more restricted graph like models (A) shown above. Then, it proceeds by following the partial ordering graph (i.e. following the topological ordering in the ordering graph).

Note that our usage of the prefix rule is similar to the algorithm proposed in [7]. However, our implementation is different from [7] because the algorithm in [7] considers loops in the resulting model and we do not. Also, we consider the prefix set merging rule while the algorithm in [7] does not.

Let X and Y be two steps and let p represent a sequence of steps. If a path XpY is in the log, it is possible that in order to execute Y, X has to be executed somewhere before the process. This type of *cross-steps dependency* can complicate process mining. In our work, we do not consider step dependency other than that imposed by the prefix rule. We avoid the step dependency problem by carefully designing our process steps.

V. Designing The Process Steps

The most challenging aspect of process mining is the design of the process steps. Each step essentially is a Python script that applies some manipulations of the current data and passes the resulting data to another step (except for the last step).

The importance of the step definition is similar to the importance of *feature selection* in machine learning. It has been widely shown that while the learning algorithm matters, the features to define the learning space can significantly impact the learning result. A notable recent example is the deep learning network [8] where much of the computation is for learning the importance of features.

Steps in process mining are like features. Hence, the design of the steps is a crucial part of the overall learning approach. An important stage for the development of the work is to recognize that in correlation analysis, the end result is basically a figure as shown in Fig. 8.

Fig. 8 is important because it provides a view to design the steps. In this view, steps can be divided into 7 categories:

- Defining the meaning of a dot
- Defining the population of the dots
- Defining the x axis, and defining how the x value is calculated for a dot (two steps)
- Defining the y axis, and defining how the y value is calculated for a dot (two steps),
- Optionally, defining how the dots are classified into two colors.

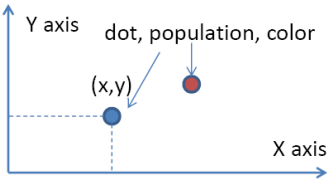

Fig. 8. Dimensions to consider in designing process steps

These 7 categories of steps basically define how the dataset should be constructed to generate such a plot. Then, two additional categories of steps are defined: (1) what analysis tool to apply, and (2) what type of result is to be reported (e.g. correlation value $>$ 0.75). The following provides an example list of process steps to illustrate our software design.

- (Dot type): PA (part), WF (wafer)
- (Wafer grouping): WBTH (binary grouping using a threshold on the number of fails from the wafer).
- (Population (lot/wafer)): LS (select all lots), L1S (select a lot), LSS (subset of lots), LSW (subset of wafers).
- (Population (dynamic)): CD (cluster lots by date), CLY (cluster lots by yield), CWY (cluster wafers by yield).
- (Population (location)): RR (restrict to a ring on wafer), RC (restrict based on a radius from the center of wafer).
- (X/Y axis (test)): X-TS/Y-TS (select all tests), X-T1S/Y-T1S (select a test), X-TSS/Y-TSS (select a subset of tests), X-T1B/Y-T1B (select a test bin), X-TSB/Y-TSB (select a subset of test bins).
- (X/Y axis (E-test)): X-P1S/Y-P1S (select an E-test from a site), X-PAS/Y-PAS (select an E-test from all sites), X-PSS/Y-PSS (select an E-test from a subset of sites).
- (x/y value (test)): X-1V/Y-1V (use the single test value), X-AV/Y-AV (use the average value), X-WNF/Y-WNF (use number of failures), X-WST1/Y-WST1 (use a statistics of the distribution), X-WNV1/Y-WNV1 (use the number of parts having a particular test value), X-WNV/Y-WNV (use the number of parts having a value in a group of test values),
- (XY coloring): X-TH/Y-TH (binary coloring based on dot property, e.g. pass/fail, or a threshold)

Three analysis tools are included: (1) statistical correlation (SC), (2) heatmap association (HA) (e.g. Fig. 4), and (3) association analysis (AA) (e.g. Fig. 5). For example, the table below shows the traces to produce the figures shown above.

Figure produced	Trace taken to produce the dataset
Fig. 2 (left)	[WF, LS, X-PAS, X-AV, Y-T1S, Y-WNV1, SC]
Fig. 2 (right)	[WF, LS, X-PAS, X-AV, Y-T1S, Y-WST1, SC]
Fig. 3	[WF, CD, X-PAS, X-AV, Y-T1S, Y-WNV, SC]
Fig. 4	[PA, L1S, X-TS, X-WNF, Y-T1S, Y-WNF, HSC]
Fig. 5	[WF, WBTF, LS, X-P1S, X-1V, Y-P1S, Y-1V, AA]

Take the Fig. 2 (left) as an example. The first step "WF" defines each dot as a wafer. The second step "LS" defines the population comprises all wafers. Then, "X-PAS" defines the x-axis is based on values of E-test from all sites. The "X-AV" defines the x value is the average. Similarly, "Y-T1S" defines the y-axis is a test. Then, "Y-WNV1" defines the y value is the number of parts having a particular test value. Lastly, a dataset is analyzed by a statistical correlation tool (SC).

In our design, steps such as "X-PAS," "Y-T1S," and "Y-WNV1" involve implicit enumeration across all possible choices. Hence, a trace involving them constructs a number of datasets through these implicit enumerations.

Note that adding new steps to the above set is straightforward. For example, if we desire to include Canonical Correlation (CC) [3] in the analysis, we just need to include new steps to define the x value and the y value. For instance, instead of a single x value, the x is defined as a vector of values. Also, we need a new analytics step CCA.

VI. Applying PM Model

The product for the study is an automotive part that operates in the 76-77 GHz band allocated for vehicular radars on an unlicensed basis (see e.g. FCC page [9]). To meet the specification, packaged chips are tested in cold temperature by operating at 76 GHz and in hot temperature by operating at 77 GHz. In both conditions, the voltage required to drive the oscillator is measured. An upper limit for the hot testing and a lower limit for the cold testing are set for the measured voltage. Unexpected yield drop is observed on some assembly lots with the cold and hot temperature tests.

Final test data are organized by assembly lots. Using each chip's ECID, the data are reorganized into their production lots. Figure 9 shows wafer-to-wafer variations of the test values at cold and hot temperatures. There are 175 wafers arranged by their production lots.

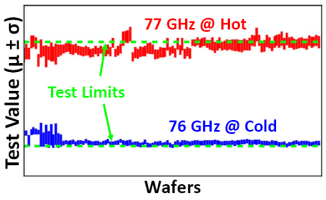

Fig. 9. Yield issue due to cold/hot voltage tests

For every wafer, two vertical bars are shown, corresponding to the cold (blue) and hot (red) results. Each bar shows the range of measured values from dice on the respective wafer. This range is $[\mu-\sigma, \mu+\sigma]$ where μ stands for the mean and σ stands for the standard deviation. The upper and lower limits are shown as two horizontal dash lines. As seen, hot values drift more frequently beyond the limit than cold values.

A. Learning a PM model

Our goal is to learn a PM model from the analytics traces conducted in work [1] and apply the model to analyze the yield issue of the automotive radar chip product line.

In the earlier discussion, we describe a *path* as a sequence of steps to construct a dataset. In the actual work, a *trace* can be a concatenation of multiple paths. For example, the first path is used to narrow the search to a particular test and then the subsequent path is used to explore the temporal aspect (as discussed above) based on the test.

The extension of a trace to comprise multiple paths requires defining a few additional conjoining steps. For example, the step HRY restricts the choices of y to the selected datasets from the previous step. Similarly, the step HRX restricts the choices of x. And HR the step restricts choices of both x-axis and y-axis. These steps contain an evaluation for the selection, for example based on a correlation value $> t$ where t is an input parameter. Finally, a general step PLOT is implemented to generate a plot.

For the result presented in this section, the log contains 39 traces. A trace can contain 1, 2, or 3 paths. Different prefix lengths are explored and results are shown in the table below.

Prefix length l	0	1	2	3	4	5	6
Number of traces	98990	1271	160	63	53	42	39

As the table shows, for an l-prefix rule, the larger the l is, the smaller the number of traces is contained in the resulting PM model. At $l = 6$, there is no generalization. For $l \leq 1$, the numbers of traces are large, which might be considered as over generalized. One can select a perceived reasonable model to apply. For example, this can depend on the desirable runtime - a more generalized model would run slower because there are more traces to execute. In our case, we use the model based on the 2-prefix rule.

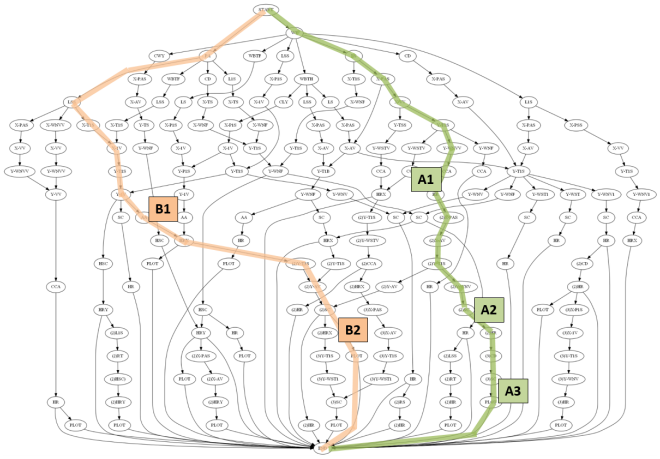

Fig. 10. 2-prefix PM model learned from the 39 traces

Let S_a be the set of traces in the l_a-prefix PM model, and S_b be the set of traces in the l_b-prefix PM model. Note that the property $S_a \subseteq S_b$ holds true if $l_a > l_b$.

The 2-prefix PM model learned from the 39 traces is shown in Fig. 10. Two traces are highlighted. The first trace has three paths, marked as A1, A2, and A3. This is the trace that leads to Fig. 3. The three paths are explained below:
(A1): This step uses CCA to identify that there is a high (CCA) correlation between a test A and an E-test PP5. (A2): By restricting to test A, this subsequent step determines that the number of fails due to test values X_1, X_2, X_3 is highly correlated to PP5. This analysis is through standard statistical

correlation (A3): This step applies a temporal consideration (i.e. step CD) to uncover the result seen in Fig. 3.

The second trace is a new trace generalized by the process mining. The trace has two paths, marked as B1 and B2. This trace was a successful trace for analyzing the yield issue discussed in this section. Their results are explained through the two plots in Fig. 11.

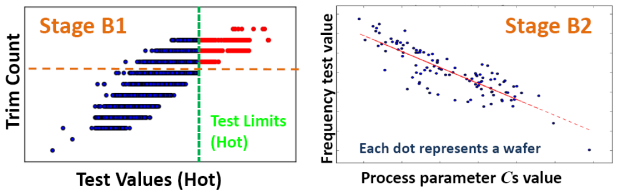

Fig. 11. Finding association(left); Finding statistical correlation (right)

The left plot shows that hot pass (blue) is associated with the variable trim count. Every dot is a part. Before the cold/hot final test, the frequency of an on-chip oscillator is measured at room temperature. Then, the oscillator is tuned by a *trim* process. Trim count is treated as a test value. In stage B1, *association* analysis based on two tests is conducted. The plot shows that all dies passing the hot test have a low trim count.

Because the trim count and the frequency test are known to be associated, in the subsequent path B2, frequency test is shown to be highly correlated to an E-test C. This correlation is shown in the right plot. Note that the direct relationship between the trim count and E-test was also explored, but it did not find any meaningful result.

The trace B1→B2 is new because in [1], we never consider association analysis using two tests. What we considered was the association analysis shown in Fig. 5 using two E-tests. However, the learning is able to generalize to include association analysis using two tests.

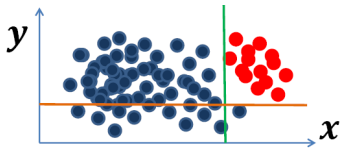

Fig. 12. Association by finding two lines

To be more specific, the association analysis can be thought of as finding two lines in a 2D space as shown in Fig. 12. In this plot, suppose red dots are failing dies and blue dots are passing dies. If one can find a vertical line such that one side of the line contains only (or mostly) one type of dots, then one can say the type of dots is associated with the x variable. Similarly, finding a horizontal line can decide if there is an association to the y variable.

In the left plot of Fig. 11, the red type is associated with the hot test. However, this is not useful because failing dies are decided by the test. The blue type is associated with the trim count. This is useful because that indicates all passing dies have a lower trim count.

VII. Conclusion

This work presents an approach for learning the process of correlation analysis. We present a process mining (PM) algorithm and explain the design of the process steps to enable the learning. The effectiveness of our approach is demonstrated by applying the PM model resulting from the work in [1] to resolve a yield issue in a new (and different type of) automotive product line. While the result is encouraging, generalizing the approach to other application scenarios may require further enhancement of the current PM design.

As mentioned before, one limitation of the current PM approach is that we do not consider cross-steps dependency. We avoid dealing with such a dependency in the PM algorithm by taking special consideration in designing the process steps. However, the consideration also constrains the design of the steps. For other applications, the PM algorithm may need to be enhanced to explicitly take cross-steps dependency into account. In addition, our PM approach does not allow loops in the process. Considering loops will drastically increase the complexity of the PM algorithm. However, it can further relax the constraint on what types of steps we can define.

In general, there is a tradeoff between the objective to simplify the PM algorithm and the objective to allow flexibility in designing the process steps.

The capability of a PM model is limited by the set of process steps. How to learn to refine and enhance a set of process steps can be another interesting future research question.

Furthermore, how applicable a PM approach is to other types of analytics applications in test remains questionable. For example, if for an application the main challenge in Fig. 1 is not with the dataset preparation box but with the meaningfulness determination box, then further research is needed to determine if the meaningfulness determination requires a complex process or not. If the process is simple, then applying the PM approach might not make sense.

References

[1] Jeff Tikkanen, Sebastian Siatkowski, Nik Sumikawa, Li-C. Wang and Magdy S. Abadir. Yield Optimization Using Advanced Statistical Correlation Methods. International Test Conference, 2014. 10 pages.

[2] Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, 2011, pp. 2825-2830.

[3] David R. Hardoon, Sandor Szedmak, John Shawe-Taylor. Canonical correlation analysis; An overview with application to learning methods *Neural Computation*, 16 (12), pp. 2639-2664, 2004.

[4] R. Agrawal, D. Gunopulos, F. Leymann. Mining process models from workflow logs. Proc. 6th International Conference on Extending Database Technology, 1998, pp. 469483.

[5] W.M.P. van der Aalst, A.J.M.M. Weijters, L. Maruster. Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 2004, pp. 11281142.

[6] Dana Angluin and Carl H. Smith. Inductive Inference: Theory and Methods. *ACM Computing Surveys*, 15 (3), 1983, pp. 237-269.

[7] W.M.P. van der Aalst, V. Rubin, H.M.W. Verbeek, B. F. van Dongen, E. Kindler, C.W. Gúnther. Process mining: a two-step approach to balance between underfitting and overfitting, Software System Model, 9, 2010, pp. 87-111.

[8] Yoshua Bengio, Ian Goodfellow, Aaron Courville. Deep Learning. book in preparation for MIT Press (http://www.deeplearningbook.org/)

[9] https://www.fcc.gov/document/operation-radar-services-76-81-ghz-band