PACO: A Predictive Analysis System for Manufacturing Test

Timothy Jones
Adaptive Systems, GS Test Engineering
Rockwell Collins, Inc
Cedar Rapids, IA
tbjones1@rockwellcollins.com

Joseph Engler
Adaptive Systems, GS Test Engineering
Rockwell Collins, Inc
Cedar Rapids, IA
jjengler@rockwellcollins.com

***Abstract*— Testing of products throughout the manufacturing process results in a vast amount of data which is often analyzed in a post process model. Analysis of this data is typically limited to standard Six Sigma or statistical process control methodologies. This form of analysis is susceptible to overlooking correlations between differing test configurations as well as other non-intuitive relationships within the data itself. Often, this analysis is performed too late in the testing process to affect the most beneficial change. Post processing of test data requires that complete testing be performed even though failures may occur during the testing process. This paper presents a predictive analysis system which analyzes test data in a near real-time environment. The presented Predictive Analysis Collaboration Object (PACO) receives real-time data from the testing system and performs data analysis against a constantly evolving rule set to offer rapid feedback to the appropriate personnel. Thus, the user has the ability to stop testing or prevent the unit from entering another test setup phase should faults be detected. Illustrated within this paper is the basic architecture of PACO along with the analysis algorithms which are currently utilized in the analysis. Additionally, a case study of PACO in use on a product under test at Rockwell Collins Inc. is given.**

Keywords-predictive analysis, test data, test station

I. INTRODUCTION

Testing of assemblies in the manufacturing process produces vast amounts of data. Often this data is analyzed in a post process model through the use of Six Sigma or other statistical process control methodologies. It is widely accepted in industry that data analyzed in a post process model is suboptimal. Real time analysis is considered to be more beneficial to the testing and evaluation process.

Real time data analysis facilitates effective fault prediction. Fault prediction is well represented in the literature. As examples, Maly [1] described fault modeling for VLSI testing. Kim *et al.* [2] suggested using historic data to predict faults. Catal *et al.* [3] introduced a predictive model that utilized limited data for prediction. Ostrand *et al*. [4] described metrics to measure fault prediction success.

Largely missing from the literature is the utilization of real time data to produce real time fault predictions during manufacturing test. This paper presents a predictive analysis system which analyzes test data in a near real time environment. The presented system is called the Predictive Analysis Collaboration Object (PACO) and performs analysis of test data in both a real time and offline fashion to facilitate optimal prediction accuracy. The architecture of PACO is presented as well as the algorithms used in the analysis engine of PACO. A case study of PACO in use on a subassembly under test at Rockwell Collins, Inc. with a simulated test run is also given.

The remainder of this paper is organized as follows. Section 2 provides a system overview of PACO. The analysis algorithms are discussed in Section 3. The presentation of a case study is given in Section 4 and conclusions are given in Section 5.

II. PACO SYSTEM OVERVIEW

PACO is a predictive system designed in a modular format to accommodate multiple testing scenarios. The purpose of PACO is to capture data from a test station (or multiple test stations) and use that data in analyses to form fault predictions. PACO has the ability to report predicted faults to the end user during test, enabling an expedited repair. Additionally, PACO's capability spans assembly boundaries by allowing correlation between measurements from a subassembly test to indicate failure at the top level assembly even when the subassembly tests may not fail. There are five basic components that make up the PACO system. These components all work in conjunction to collect, store and analyze data from the test station and report predictions and trends to the end user. These components are described below.

Test measurements and results from the test station are stored in the PACO database through the PACO interface module. Instantiated at the onset of the test run, the PACO interface launches a PACO listener module as well as facilitating the storage of the test data to the PACO database. This interface is a simple .NET dynamic link library (DLL), which can be utilized by any compatible system. At Rockwell Collins, Inc. this agent is used in conjunction with a test architecture that includes National Instruments TestStand©.

The PACO listener performs a registration with the PACO server, creating a unique subscription representing the test instance. This transaction allows for the identification of a running test instance in the system as well as establishing a subscription to predictions that may be posted to the server.

978-1-4244-7961-0/10/$26.00 ©2010 IEEE

This subscription provides an interface for prediction event notifications.

The PACO server acts as a message server between the PACO listener on the test station and the PACO runtime monitor, maintaining a library of connected and active modules in the system. A PACO runtime monitor can post messages to the server, which will be relayed to any listener modules that are subscribed to the same subscription. This server also facilitates the logging of prediction events to both the PACO database and an internal activity log.

PACO also has an inference engine which performs analysis on the historic data collected by PACO. Various algorithms are incorporated into the PACO Inference Engine (PIE) including decision tree type analyses and Apriori market basket style algorithms. These types of algorithms generate rules that are stored in the SQL database for comparison with the incoming data from the active test stations. Rules from these types of analyses take the form: "*if measurement 11 is high, top level testing will fail with 98% confidence*" or *"if measurement 5 is low, there is a 87% chance of test 12 failing*". Rules generated by PIE are only saved if they are determined to be significant (within a given threshold of confidence) and novel. Users do have the ability to manually remove rules from the rule set if they are determined to be trivial.

The SQL database facilitates the storage of all centralized data for the system. This includes the collected test data deposited by the PACO interface, any predictions made to the end user by PACO and rules discovered by the algorithms utilized by the PIE.

The PACO runtime monitor evaluates the runtime data against any rules that have been established by PIE for the running configuration. Additionally, the PACO runtime monitor detects standard Six Sigma trends in the data such as 6 increasing or decreasing points. The PACO server performs the assignment of inactive subscriptions to the PACO runtime monitor as well as notifying the monitor when the subscription is no longer valid. Should real time data come in that matches a high confidence rule, the PACO runtime monitor posts an alert to the PACO server, notifying the end user of the possible fault detected. Alerts sent to the users take the form: *"measurement 8 was high therefore the likelihood of passing top level testing with this module in the assembly is only 2%"*. Rule alerts such as this allow the end user to make knowledgeable decisions about the assembly and thus reduce the time required to identify and correct failures as well as reducing waste created by running test configurations that are known to fail.

Fig. 1 below illustrates the system components of the PACO system. It is clearly seen that these components would be able to effectively work together to inform the test technician of any issues within the test run being performed.

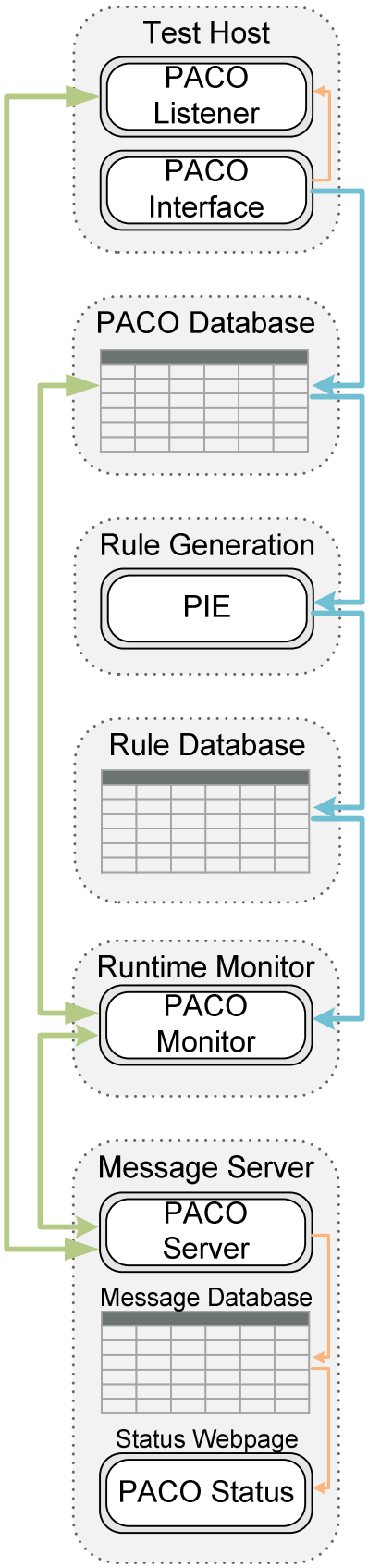

Figure 1. PACO System components

Through the modularity of the system's design, PACO affords the ability to customize an installation to multiple system configurations. The PACO interface and listener are designed to exist in a resource-starved environment, allowing for an increase in data analysis with a minimal impact on station resources. This modular form, with test station efficiency in mind, allows PACO to be implemented on a single station or on a network of stations with external servers dedicated to the processor-intensive data analysis.

III. Analysis Algorithms

PIE forms the heart of the PACO system and produces all of the rules used by the system for analysis of the real time data. PIE runs at configurable frequency allowing for the system to be updated on a predetermined schedule. This section describes two of the algorithms that are associated with PACO. It is important to note, however, that PACO is easily extended to include other analysis algorithms as well.

One of the rule-based analysis algorithms used by PIE is the decision tree algorithm. Decision tree algorithms are useful data mining algorithms which generate rules based upon a classification split [5]. A decision tree is comprised of nodes in a hierarchical fashion such that there exists a top level node (known as the first split) which contains children nodes. Children nodes can be further split or they can be terminations (also called leafs).

Splits within the decision tree are based upon the current set of data, queried from the historic data in the case of PACO. The data is analyzed to determine which variable offers the cleanest split. In test, this split is based upon passing or failing a test. In this circumstance, the cleanest split is defined as the fork in the result data which provides the most informative branch in the decision tree. Thus, if the current dataset indicates that 40 of the test runs failed *measurement 12* when it trended high and that 50 test runs failed *measurement 13* when it trended low, measurement 13 would be used for the split due to a more conclusive split information. The 50 test runs that failed on *measurement 13* when it trended low are then removed from the dataset and a leaf is added to the current node stating that *"when measurement 13 trends low the test run will fail*". The remaining dataset will then be evaluated in an iterative manner to generate the remaining hierarchy of the tree.

It can be seen that, should the decision tree algorithm produce rules which are novel (non-trivial), reduction in testing time could be achieved. This is especially true for testing which involves many cycles or lengthy test times. The ability for the decision tree to state, with a given confidence level, that a test will fail allows the user to react in an appropriate manner and halt testing in a timely fashion to repair the faulty assembly.

Decision tree rules are important for discovering tests which, when trending in a specific manner, cause the unit under test (UUT) to fail. The decision tree analysis allows for users to save on lengthy test times. The reality is that users will only follow the advice of the system if they are confident in the alert. Decision trees generate a confidence measure that can be presented to the user. This confidence metric is based upon the cleanliness of the split at the given node. The confidence metric used in PIE is given in Eq. (1) below.

$$C_i = \frac{|c_1|}{N} \tag{1}$$

where i is the node under consideration, c_1 is the number of test runs which form the leaf of the node under consideration and N is the total number of test runs.

The confidence metric is stored in the PACO database once it is calculated and is presented to the user as a measure of accuracy of the rule that was generated. Rules whose confidence falls below a given threshold, ε, are discarded as they are too infrequent to be of value to the test technician.

The second type of rule-based analysis performed in the PIE is known as association rule analysis. This type of analysis determines associations, or relationships, between tests. In the PACO system, these associations can occur within the same module level testing or between sublevel and top level testing.

For the association rule analysis the historic data in the PACO database must be discretized. This is performed by generating partitions of the data for each test. These partitions are often of the type fail low, trend low, good, trend high, and fail high. The data for each test within a test run in the dataset is then discreetly placed in one of the bins for that test (I.E. if *measurement test 8* trended high, a 1 is added to the trend high bin for that particular test. All the other bins for that test receive a 0). The discretized data is used by the association rule algorithm to generate rules for the PACO system.

The generation of association rules is an iterative process. The process begins by forming 1-item sets which are frequent in the discretized dataset. A 1-item set is a set of a single discretized bin in the discretized dataset. Thus, *measurement 10 trend high* is a 1-item set. This item set is frequent if it occurs in the database with a frequency greater than some threshold, τ. Those 1-item sets which are frequent are then combined to form 2-item sets. The 2-item sets are then evaluated for frequency and those which are infrequent are removed from the computation. The combination of item sets and the evaluation for frequency is performed iteratively until no further combinations can be made. The removal of the infrequent item sets is constrained by the Apriori principle which states that infrequent item sets cannot be members of frequent itemsets of larger size.

The association rule analysis can generate many rules that are somewhat trivial (I.E. *measurement 1* at 52Mhz trends high and *measurement 1* at 75Mhz trends high implies that *measurement 1* at 90Mhz trends high). These trivial rules are removed through domain expertise and through automated filtering algorithms. The rules remaining in the system are considered of adequate significance to monitor.

The association rule analysis is performed on data of the same module level of testing as well as upon a combination of sublevel and top level testing. These inter-module association rules create coherence between assemblies, allowing for the capture and characterization of system dynamics throughout the build process. This insight aids in early identification of failures induced through module interaction, reducing the time needed to determine subassembly incompatibility which would have only been visible at the top level assembly.

In addition to the processing of historic data the PACO system performs trend detection of standard SPC trends. Standard SPC trends (for a given test) include six or more consecutive data points increasing or decreasing, nine or more consecutive points on the same side of the median of the data, and fourteen or more consecutive points alternating up and down [6].

The PACO system performs analysis of trend detection through a rapid query and response protocol which allows for comparison of the real time data with historic data to alert the user on trends detected in each test. As real time data enters the system it is rapidly compared to recent historical data to determine if any of the standard SPC trends are present. Should a SPC trend be detected the user is notified through the PACO Server.

The SPC trend detection analysis offers a different view of products in manufacturing than the rule-based analysis. While the rule-based analysis allows for detection of UUT specific issues, the SPC trend detection allows for detection of issues that may be more closely related to the test station or component lot being utilized for the product.

The combination of rule-based and trend based analysis of the real time data allows PACO to be highly efficient at detecting problems during the manufacturing process rather than in a post process model. The added ability of PACO to include other data analysis modules into the analysis engine results in a highly customizable system as well.

IV. Case Study

The PACO system is being implemented at Rockwell Collins, Inc by the Government Systems Test Engineering group. The implementation of PACO at Rockwell Collins Inc. is in conjunction with National Instruments TestStand©. Testing is performed on automated test stations (ATE). The ATE software feeds data to the PACO interface through a customized TestStand© interface. The PACO listener is also installed on the ATE.

The implementation at Rockwell Collins of the PACO system is distributed over various computing devices. The ATE hosts the PACO interface and listener. A separate server hosts the PACO database. PIE is also hosted on a separate server to facilitate more rapids analysis. Finally, the PACO server resides on its own machine to facilitate large amounts of data traffic.

This section describes a case study of a test run of a single subassembly at Rockwell Collins using the PACO system. For purposes of disclosure the test run is simulated and the name of the product has been changed, as have the names of the tests. The product under consideration for this case study had 5559 historic test runs stored in the PACO database prior to the test run represented here. PIE had performed the historic analyses of the decision tree and apriori style algorithms prior to the introduction of this simulated test run. The test run being illustrated here encountered 4 alerts throughout the run from the PACO system.

As an alert is received by the PACO listener it is fed to the user via a dialog box which minimizes to the operating system tray on the ATE. Fig. 2 below illustrates an example of the alert dialog being displayed for a test which was trending high.

Figure 2. Example of screen alerting trend detection to user.

Fig. 3 illustrates the PACO system alerting the user that a test trended high and the impact that would have on top level testing if the subassembly is included in a top level assembly.

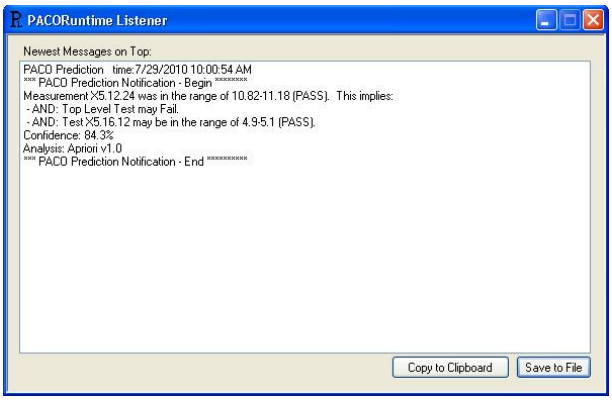

Figure 3. Association rule alert from PACO

The product under test in this case study is still in the implementation phase of utilizing the PACO system. We are unable to report yield or test time improvements that are truly attributable to the PACO system. However, it is clear that this system will directly benefit test engineering and manufacturing as it is becomes fully implemented.

V. Conclusions

This paper has presented a predictive analysis system for real time data analysis and fault detection of products under test. The architecture for this system was described in detail. The analysis algorithms utilized currently by the predictive analysis system were discussed. A case study of the PACO system on a product with a simulated test run was given.

Future work in this domain includes adding additional analysis algorithms to PIE. Nonlinear analysis is planned for incorporation into PACO. The nonlinear analysis additions will assist in detecting nonlinear trends and alert users to trends which are outside the standard SPC trend domain.

REFERENCES

[1] Maly, W. "Realistic Fault Modeling for VLSI Testing", *Proc. of the 24th Annual ACM/IEEE Design Autmoation Conf.*, Miami Beach, FL., pp. 173-180, 1987.

[2] Kim, S., Zimmermann, T., Whitehead, E., and Zeller, A. "Predicting Faults from Cached History", Proc. of the 29th *Intl Conf on Software Engineering,* Washington, DC, pp. 489-498, 2007.

[3] Catal, C. & Diri, B. "A Fault Prediction Model with Limited Fault Data to Improve Test Process", Lecture Notes on Computer Science: Product-Focused Software Process Improvement, *Springer*, Berlin, 2008.

[4] Ostrand, T. & Weyuker, E. "How to Measure Success of Fault Prediction Models", *4th Intl Wrkshp on Software Quality Assurance SOQUA '07*, New York, NY, pp. 25-30, September 2007 .

[5] Witten, I., & Frank, E. "Data Mining: Practical Machine Learning Tools and Techniques", *Morgan Kauffman*, San Francisco, 2nd Ed., 2005.

[6] Pande, P., Neuman, R. and Cavanagh, R. "The Six Sigma Way: Team Fieldbook", *McGraw Hill*, New York, NY, 2002.