IOWA STATE UNIVERSITY
Digital Repository

Creative Components

Iowa State University Capstones, Theses and Dissertations

Fall 2018

Using Predictive Maintenance techniques and Business Intelligence to develop smarter factory systems for the digital age

Tejaswini Vuyyuru
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/creativecomponents

 Part of the Management Information Systems Commons

Recommended Citation

Vuyyuru, Tejaswini, "Using Predictive Maintenance techniques and Business Intelligence to develop smarter factory systems for the digital age" (2018). *Creative Components*. 114.
https://lib.dr.iastate.edu/creativecomponents/114

This Creative Component is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Creative Components by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

www.manaraa.com

Using Predictive Maintenance techniques and Business Intelligence to develop smarter factory systems for the digital age

by

Tejaswini Vuyyuru

A creative component report submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

Master of Science

Major: Information Systems
Minor: Statistics

Program of Study Committee:
Dr. Sree Nilakanta, Major Professor
Dr. Jim Davis
Dr. Will Meeker

The student author, whose presentation was approved by the program of study committee, is solely responsible for the content of this report. The Graduate College will ensure this report is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

Copyright © Tejaswini Vuyyuru, 2018. All rights reserved.

www.manaraa.com

TABLE OF CONTENTS

Page

www.manaraa.com

LIST OF FIGURES

Page

www.manaraa.com

NOMENCLATURE

CBM	Condition-Based Maintenance
UI	User Interface
MES	Manufacturing Executive System
SAP	Systems, Applications and Products
IT	Information Technology
EDA	Exploratory Data Analysis
PLC	Programmable Logic Control
OEE	Overall Equipment Effectiveness
FPY	First Pass Yield

www.manaraa.com

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to our Head of the Committee and Major Professor Dr. Sree Nilakanta, for always being as a significant support to reach great heights and is an inspiration to set sky as the limit in attaining knowledge and face the practical side of various subjects.

My special thanks to Dr. Jim Davis, for taking time out of his busy schedule and spending invaluable time with me, thus, motivating me to put together this report in a structured format to yield best results.

My sincere thanks to Dr. Will Meeker, Minor Professor for accepting my offer to be on the committee representing Statistics as my minor subject and for his guidance and support throughout the course.

In addition, I would also like to thank my manager at Danfoss, friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.

www.manaraa.com

ABSTRACT

The aim of the project is to increase the productivity of Danfoss' assembly lines across the manufacturing facility at Ames, Iowa by continuously monitoring the performance with the help of a real-time tracking tool. The efficiency of the employees at each of the stations in the assembly lines, the time taken to procure the products or parts, the test time and the paint time, all impact the performance and production rate within the facility. Also, keeping few attributes constant, the time taken for the machines to pass a particular station within an assembly line determines the health of the assembly line and requires continuous monitoring. Minute errors on the assembly lines could stall production for hours resulting in an immense loss to the giant manufacturing companies like Danfoss.

Therefore, the project is about implementing an algorithm that will monitor both the number of workers per shift per day at the assembly lines and the status of the machines on the line as they pass through each station on the assembly lines. On the other hand, a user-interactive dashboard allows the workers to monitor their progress versus the expected progress for the day. The live dashboard is scalable to other Danfoss assembly lines as a daily monitoring system.

Keywords: *Business Intelligence, Predictive Maintenance, Analytics, Machine Learning, Manufacturing, Time series Analysis*

www.manaraa.com

CHAPTER 1. INTRODUCTION

The Danfoss Group is a manufacturer of products and services used in cooling food, air conditioning, heating buildings, controlling electric motors, compressors, Variable frequency drives and powering mobile machinery. The company is also active in the field of solar and wind power as well as district heating and cooling infrastructure. Danfoss employs approximately 24,000 people (Wikipedia, 2017) worldwide with its headquarters in Nordborg, Denmark.

Danfoss was founded in 1933 by Mads Clausen and is today almost entirely owned by The Bitten and Mads Clausen Foundation. In 2002 Danfoss joined the United Nations Global Compact, consisting of nine principles with social and environmental responsibility. The company has an annual sales turnover of 29.2 billion DKK (2016) and has sales companies in 47 countries and 56 factories in 18 countries around the world. Their distribution network for solar inverters covers 82 distributors and wholesalers across 27 different countries.

Danfoss has four segment – Power Solutions, Heating, Cooling and Drives. For a major B2B company like this, procuring raw materials, assemble them according to the needs of the customer and delivering them is very important to retain their market and create a value proposition. The production facilities are the most essential powerhouses where the products are assembled, packed and shipped to the end consumer. As a result, the maintenance of the production facility requires significant attention to make sure that everything is in place thus, ensuring smooth production activities.

www.manaraa.com

Age of Digital Transformation

Big Data Analytics has become an essential segment of every organization. Be it a manufacturing firm with operations in more than 100 countries or a start-up emerging as a future leader in the market, demands the capability to predict future events. Especially in this digital age, when enormous data is readily available, all one needs to do is to devise methodologies to process this data and derive trends which help in understanding what can be improved. Predictive Maintenance is one such field within Predictive Analytics of the four types of Analytics namely:

- Descriptive Analytics
- Diagnostic Analytics
- Predictive Analytics
- Prescriptive Analytics

Today, a large number of modern manufacturing companies are starting to incorporate data analytics and machine learning in their production and manufacturing process. For an organization like Danfoss, having Digital Transformation as one of the behaviors of its thought leaders, Predictive Maintenance is a niche area with several engineers pushing to understand the trends within the data. There are several organizations and business processes incorporating predictive maintenance thereby, not only building the capability to tell when a device failure can occur but also prevent the failure from occurring in the first place. This kind of predictive systems allows these companies to plan machine maintenance adaptively rather than on a fixed schedule. The purpose of such a system is to improve quality control, product quality and reduce costs, by forecasting equipment breakdowns and scheduling maintenance before they

www.manaraa.com

occur. Besides providing all these benefits, this would also improve the accuracy of detecting such failures and optimize periodic maintenance operations that are carried out.

Predictive Maintenance is also about developing a real-time monitoring system to strictly keep tabs in order to eliminate any possible production breakdowns. It is a cost-saving business strategy to minimize loss and increase profits. To emphasize it enables leaders to make better decisions to:

- increase/decrease the workforce during the shift,
- analyze the number the production systems or assembly lines used for production
- manage inventory
- plan delivery
- build robust logistics and manage supply chain
- study the performance and to improve the efficiency of a system

www.manaraa.com

Scope and Outcome

This project consists of two parts, where the first part is to analyze the data to identify faults patterns and the second part is to develop and design a user-interactive and real-time monitoring system to study the performance of the production lines. The future scope of this project is to feed this analysis into a system which will learn over time to predict the best estimation of Time to failure for the assembly line, based on early identification of unexpected events. This will allow the company to repair the machine before it breaks, saving cost and minimizing machine downtime.

The final outcome is to have:

- A design framework to consume existing assembly line data
- A business intelligence framework for dynamic display of trends in data
- A real-time on the floor monitoring system to examine the efficiency and performance of the assembly lines

www.manaraa.com

CHAPTER 2. LITERATURE REVIEW

Having a dynamic system to monitor business process across production facilities continuously results in eminent maintenance. As a result, there is an increased performance which further reduces the repair cost drastically. Given the advantages of incorporating such intelligent systems in the production facility, organizations are tremendously investing in research and development associated with improving the industry best practices.

Need for Predictive Maintenance within Industries

The fundamental purpose of maintenance of any business is to provide the required capacity for production at a lower cost. It should be regarded as a "reliability" function and not as a repair function (Beebe, Ray S. 2004). Predictive Maintenance or condition-based maintenance (CBM) is an attempt to forecast the imminent occurrence of a failure to assist an intervention and therefore, preventing the failure in the first place (Eisemann RC. 1998). This predictive maintenance has resulted in minimizing the cost of maintaining the machines in production facilities, improve operational safety and reduce the quantity and severity of in-service machine failure. (Rao, B. K. N. 1996).

On the other hand, Big Data platforms provide a scalable, robust and low-cost option by deriving insights from the historical data (Mohanty, S.; Jagadeesh, M.; Srivatsa, H. 2013). Predictive Maintenance which is a subset of the Big Data systems provides a holistic approach to study the business process and accelerates the decision-making process. Hence, a diagnosis of the fault of an assembly line when scaled at a production facility level should be coupled with predictive decision-making systems.

www.manaraa.com

Business Intelligence to provide predictive maintenance

The primary aim of shifting the focus toward Big Data especially in the manufacturing industry is to minimize the downtime cost and improve efficiency and performance. Deloitte, a giant consulting firm, through its market research, confirmed that implementing advanced analytical processes results in about 10 - 20% increase in the equipment uptime and availability (Deloitte Consulting, 2017). Although the results of Predictive Maintenance are far fetching, robust methodologies and business models should be developed to yield insightful outcomes.

The foremost required to designed strong business models for predictive maintenance is to have a continuous stream of sensor data also known as time series data which records the time at which the operations take place. The entire data is a series of timestamps collected over a period of time (Perera, Srinath; Alwis, Roshan; 2017). The time series data is implemented in Business Intelligence (BI) systems to understand "what has happened" followed by deriving actionable insights about "what can be done." There it is utmost important to understand the "what" portion before arriving at the step to prevent an event from occurring in the first place as in predictive maintenance. (Walker, Peter; ComputerWeekly; Information Builders; 2011).

www.manaraa.com

CHAPTER 3. RESEARCH DESIGN

Danfoss has a robust system to generate, store and process the production data. The company has sensors on machine lines, which monitors and collect data timely (putting a timestamp on it). The data is continuously generated and stored on the SAP system within the production IT domain. This data collected from the SAP system consists of several tables which require to be aggregated and merged together. This data should be further linked to the sensor data to obtain the time stamp data for assessing the time taken for the products to pass through each station on the assembly line. This paper addresses the general challenges of obtaining data access, aggregation, and preliminary trend analyses and developing requirements.

System Specifications

The end system created has the following functional requirements:

- Analytics Dashboard: A real-time user-interactive dashboard is built. This dashboard is placed on a screen in the production facility to deliver live updates across the entire facility by tracking the performance and efficiency across the assembly lines
- Predictive analytics: The dashboard system can also analyze the data and within a degree of certainty, predict if the performance is lower or higher with respect to historical data
- Historical analytics: The system also displays the varying trends throughout a particular shift for any historical data

www.manaraa.com

Methodology

The data required to initiate the analysis is extremely sensitive from the organization's point of view. It is important to get access to the data by following a Danfoss protocol of taking the IT security training and then requesting to be mapped to the server. The dashboard should be developed by utilizing the existing Danfoss tech stack, namely Industrial Automation's Ignition. This solution provides the basis of the User Interface (UI) and displays functionality, the data should be fed into the system by extracting the required tables from the server. Data will be gathered and sanitized from the existing Manufacturing Executive Systems (MES) database structure wherein Danfoss stores all relevant product line statistics.

All the assembly lines have sensors across each of the stations in the production facility. These sensors emanate data continuously with a timestamp assigned to each product that passes through the stations. This real-time data is stored in several fragments within the MES on the production IT. There are about ten different tables which different data pertaining to the product type, model, specification and time. The data from all these tables are aggregated into a single large dataset to carry out the initial descriptive analyses to understand the data. The various types of initial analyses were to visualize the trends in Excel to find the outliers and handle missing values. major

The design features of the dashboard had to be determined to answer several questions which enable data-driven decision making feasible. The dashboard should be insightful for the leadership team. Therefore, several discussions and brainstorming with a wide range of stakeholders have been made as a priority in the pipeline.

www.manaraa.com

CHAPTER 4. DATA DICTIONARY

The data is stored in several fragments on the SAP server controlled by the production IT of Danfoss. The data has to be linked to the Ignition server on which the real-time dashboard is required to be developed. However, before linking the data, there is an utmost need to study the data and understand the structure. As the final dataset should be created by combining several attributes of all of these small fragments, the data should be studied for primary keys on which all of these datasets should be merged.

The following information is retrieved from all the individual tables:

- The plant code of the production facility code. For instance, the plant location has a code “1501”
- An ID which is linked to a product description namely H1 Tandem or H1 single, S90 Pump or S90 motor and so on
- A line number stating the type of product to be assembled or tested in a particular line. It also gives us the information about the paint lines of a specific product
- A log data file consisting of a unique “Serial number” and a “Pallet Serial Number” along with the time stamp data. The serial number is associated with the product and the pallet serial number is the container on which the product moves through various stations of the assembly line
- A called procedure code describing the action that is performed on the product
- A station code along with a cell id and a pass or a fail flag

www.manaraa.com

- A component ID and a component structure explaining the parts which are used in all the steps of assembling the final product. It also describes the part and links it a specific part ID
- A genealogy data explaining the product class and the parent product. It also has the information to link back to the pallet and the product serial number
- A customer code stating the entire customer information. It also gives the price of the final product and the number of products ordered along with the estimated delivery date. This information supports in keeping track and planning the production of a particular product order

Most of this information is extremely sensitive and sharing any information containing the exact column names or the attribute names does not comply with the security of Danfoss. Also, any alterations to the datasets in the server could stall production for days. Therefore, an extract of the original data on the server has been created for the initial testing phase until a final sign-off has been obtained.

Data Generation and Structure

The stations are serially aligned on the assembly lines in the production facility. As the parts are assembled to form a final product, the pallet moves on from one station to another. If a pallet is being processed at a station and the pallet in the previous has been already released, then it waits until the pallet at the current station is entirely processed and released. This amounts to the wait time of that model or product at that station. This wait time is arithmetically increasing especially when a product reaches the test station. There are two stations to test the front part of the product and the rear part of the product. In addition, by the way, the stations are designed on an assembly line, it is given that the test times are enormously high and so are

www.manaraa.com

the wait times. The following flow of the stations gives a better picture of how the stations are designed.

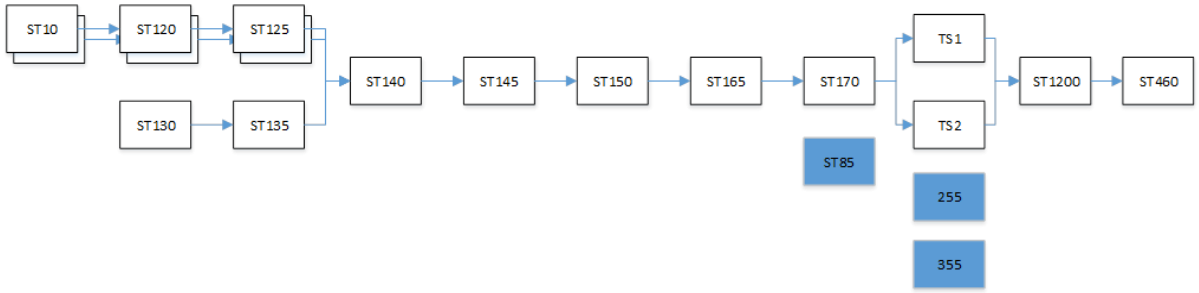

Figure 1: Design of Stations and Test Stands in an Assembly line

The parts of a product are assembled one after the other sequentially starting from Station 10 and ending at station 460. The product on an assembly line is first made in sections as shown on the diagram below:

Figure 2: Sections of a product on the assembly line

FRONT and REAR elements are assembled one after another on stations 10 through 125; the CENTER is assembled on stations 130 and 135. In the next step, on station 140 all 3 elements (FRONT, CENTER and REAR) are put together and from there they run through stations 460 until the end as one piece. The station 460 continues into the paint line which is fully automatized with the help of robotic machines.

www.manaraa.com

There are interim stations between station 10 and station 120 with the notation as station 1110. Similarly, the interim station between station 120 and 125 is denoted as station 1120. These are the wait stations where the pallet waits until the pallet in the front is completely processed and moved. Work times on stations 10, 120, 125 can be calculated by subtracting time stamps of 1110 and 10, 1120 and 120, 1125 and 125 respectively. The wait times between stations 10 and 120 and 125 are calculated by subtracting timestamps between 120 and 1110 and 125 and 1120 respectively. In similar fashion, the work times on stations 130 and 135 can be calculated by subtracting timestamps of 1130 and 130 and 1135 and 1135 respectively.

FRONT, REAR and CENTER pieces wait time is calculated as timestamp difference of 140 and 1125 for both FRONT and REAR (they are separated into the first two rows of each unique serial number). Also, the FRONT piece is in the first line and REAR piece is in the second line and they have different queue times. The CENTER piece wait time is calculated as a difference of 140 and 1135. The work time in station 140 is calculated as a difference of 1140 and 140. Now the work progresses in the very same fashion all the way to station 170.

After station 170 we have an entry 85 where station 85 is not a physical station, but rather a time at which a unit completes air tightens test at station 170. So, work time in station 170 is calculated as a difference between 85 and 170. At this stage, the data is merged with the test stand database. Unique serial numbers can be matched between the two test stands datasets and the station datasets. Test stands have entries indicating a test start time. Thus, the wait time after completion of work at station 170 (timestamp 85) is calculated by subtracting the test stand start time from the time stamp at the station 85. Test stand also gives us elapsed time of a test duration. Some serial numbers tests are re-started and those are shown as FAIL status in the results. The failed test time stamp is recorded at the time stamp of station 255. It could take

www.manaraa.com

several test iterations before the product has a PASS flag and it shifted to the paint line. As a result, the time stamp at station 255 records only last unsuccessful test event. Therefore, if a unit is tested for 4 times before it finally makes it through, the time stamp of the 4 attempts is recorded at station 255. Test completion is calculated as test start plus elapsed time.

After completing the testing phase, the unit is moved to station 1200. The duration of the process of product moving from test stand to station 1200 can be calculated as a difference between of the time stamp between station 1200 and the calculated time of a successful test completion. As a final step, the product passes through station 460. This station marks the end of work on station 1200 and the product has consecutively completed assembly and test of one tandem pump unit before proceeding to the paint line.

As a product moves through each of these stations, the sensors placed at the respective station records the event of time at which the assembly of the product is completed. The product then moves from one station to the next as shown in Figure 1. The data is real-time sensor data which is recorded sequentially as the product moves from station 10 to station 460.

www.manaraa.com

CHAPTER 5. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a significant step in the whole process. The final dataset created by joining various datasets should be analyzed and studied for outliers and missing values. Also, the trends in the data should be studied to identify the logic behind the data and further plan our next steps in the process. In addition, EDA helps in identifying the critical features of the dataset like treating the missing values, detecting anomalies within the data and removing the outliers. As discussed earlier, the dataset consists of timestamp values across all stations of the assembly line as shown below:

Model	Serial No	PALLET	10	1110	120	1120	125	1125	130	1130	135	1135	140	ID1140	ID145	ID1145	460
8302413	A181049207	013	3/14/2018 10:49:48 AM	3/14/2018 10:50:43 AM	3/14/2018 10:51:27 AM	3/14/2018 10:54:04 AM	3/14/2018 11:03:33 AM	3/14/2018 11:04:51 AM						3/14/2018 11:12:08 AM			
8302413	A181049207	021	3/14/2018 10:51:40 AM	3/14/2018 10:52:53 AM	3/14/2018 10:54:18 AM	3/14/2018 10:55:25 AM	3/14/2018 11:05:13 AM	3/14/2018 11:06:30 AM					3/14/2018 11:10:01 AM	3/14/2018 11:12:08 AM			
8302413	A181049207	026							3/14/2018 10:55:53 AM	3/14/2018 11:00:04 AM	3/14/2018 11:04:16 AM	3/14/2018 11:06:28 AM	3/14/2018 11:10:07 AM	3/14/2018 11:12:09 AM			
8302413	A181049207	BASE													3/14/2018 11:18:53 AM	3/14/2018 11:21:29 AM	3/14/2018 1:02:53 PM
8302413	A181112446	011	3/14/2018 10:56:55 AM	3/14/2018 10:57:19 AM	3/14/2018 10:58:03 AM	3/14/2018 11:01:00 AM	3/14/2018 11:13:04 AM	3/14/2018 11:14:24 AM						3/14/2018 11:23:42 AM			
8302413	A181112446	023	3/14/2018 10:58:34 AM	3/14/2018 10:59:41 AM	3/14/2018 11:01:22 AM	3/14/2018 11:02:06 AM	3/14/2018 11:15:07 AM	3/14/2018 11:16:14 AM					3/14/2018 11:22:01 AM	3/14/2018 11:23:43 AM			
8302413	A181112446	035							3/14/2018 11:07:20 AM	3/14/2018 11:09:58 AM	3/14/2018 11:19:04 AM	3/14/2018 11:21:42 AM	3/14/2018 11:22:04 AM	3/14/2018 11:23:44 AM			
8302413	A181112446	BASE													3/14/2018 11:26:41 AM	3/14/2018 11:28:45 AM	3/14/2018 1:14:38 PM

Figure 3: H1 Tandem Time at Station XX 3-14 Shift 1

Missing Values Treatment

The final dataset consists of several empty values. There is no timestamp value for the stations for some of the model numbers. This can be interpreted as that the product or the unit has been built during the time frame of the previous shift. Also, there could be a scenario where a product was starting during a particular shift, but the testing was carried onto the next shift. Consequently, the product completion time would be recorded in the next shift and the timestamp would not be recorded for that station in that shift. If this scenario occurs for more than two stations sequentially, then the time differences would be zero.

Further, the time recorded at the test stands are sometimes corrupted during a failed test. Thus, there could be entries of the format “23:57:###” which are most likely to be ignored. These values should be replaced with zero for our calculations.

www.manaraa.com

Anomaly detection and Outlier Treatment

The dataset consists of work times and wait times. Some of the work times could be exceptionally low and on the other hand, some of the wait times could be exceptionally high. This difference occurs in a case where a product or unit is being tested around lunch time. The workers on the test stand should have their lunch during the designated time frame. Hence, the product is left to stand there throughout the whole period while the unit behind it on the assembly line has already been waiting for a while to be tested. As a result, when all these time differences are converted into seconds for our analysis, we can see that the data is greatly skewed to the left.

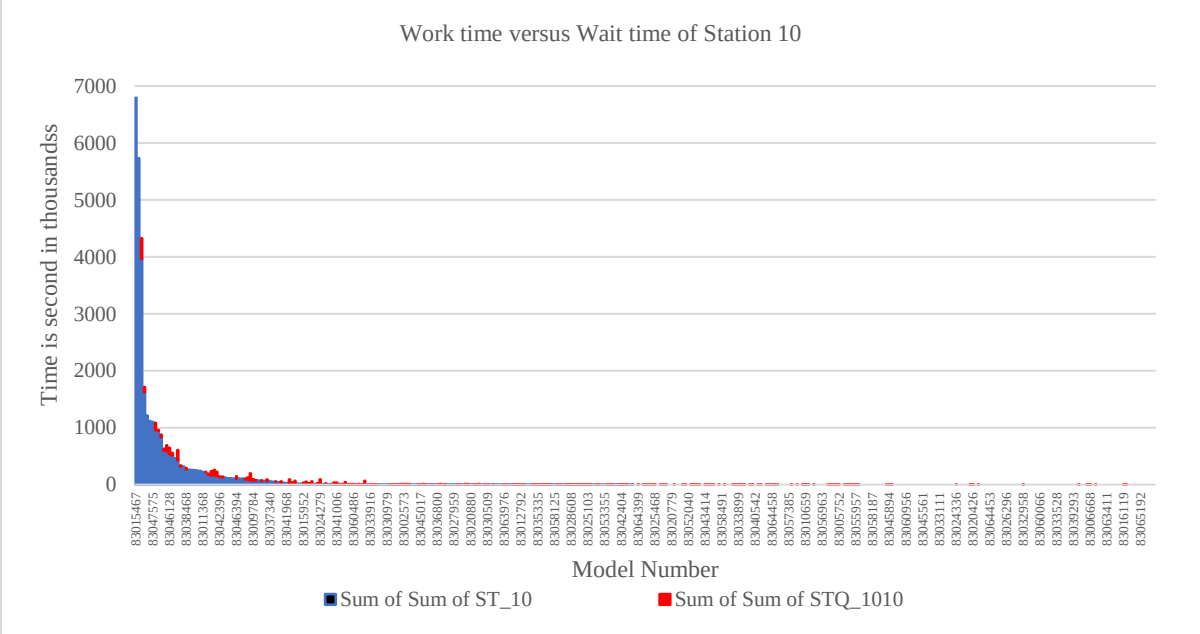

Figure 4: Work time and Wait time of Station 10

Daily values start with the first shift from 6:00 AM to 2:00 PM. Followed by the second shift from 2:00 PM to 10:00 PM and the third shift from 10:00 PM to 6:00 AM the next day. These shifts do not account for the lunchtimes and the break times and the sensors are active during this time as well, continuously recording the time for a unit at a station in an assembly

www.manaraa.com

line. Thus, this time gets added into either work time or wait time. Subsequently, this time should be removed from the timestamps depending on the data. As a consequence, the skewness is reduced by a great extent.

Moreover, the data should follow a normal distribution to analyze the average time taken by a product across each station. The skewness of the data is removed as outliers and the data is normalized. Anything outside this normal distribution would be considered as an outlier. To normalize the data, it is therefore required to anchor the distribution on the low end and mean-time and ignore the remaining portion of the data where the time differences are too low for analyses. This will set expectations of high performance across the assembly lines.

The data is not analyzed for stationarity or correlation as the primary focus is to clean the backend data to be able to improve the quality of the dashboard. So, most of the data is manually cleaned by identifying the extreme time difference between the stations. These outlier values are eliminated from being considered in the pool of acceptable time differences where the dashboard displays a red color.

The performance estimation and time difference information entitle the engineers to detect any anomalous behavior across the assembly line. Any time difference value significantly higher than the average work time or wait time of a station from the historical data must be carefully studied and the insights should be a part of the continuous improvement of the production facility. The main reason behind studying the historical trend is because the data is highly auto-correlated, implying that the present values largely depended on the past values. As a result, there was a significant impact on the time series data accommodating explanations to either addition staffing during a shift or increasing the assembly lines in the facility or to have additional test stands.

www.manaraa.com

CHAPTER 6. REAL-TIME IGNITION DASHBOARD

After obtaining a complete understanding of the data and the data dictionary, a data connection is made from the Ignition Server to the SAP server to extract all the required datasets. As a result, the entire data is available real-time, thus, eliminating the process of storing the data on another platform. The production numbers are taken from the queries and the data is loaded into the Ignition server. As discussed, the MES data earlier, this data will update each time a trigger is issued from a Programmable Logic Control (PLC).

Design Features

The dashboard is real-time and is displayed on a screen which is installed in the production facility for continuous monitoring. The system has the following non-functional features:

- Performance: The system can dynamically update and respond to changes in the data
- Security: The system is secure as it is interacting with Danfoss's production database. Also, the access is restricted to only a few engineers working on the project
- Stability: The system is stable and can withstand downtime due to production malfunctions and there was a maximum uptime to continue running analytics on the assembly line
- Scalability: The system is scalable to many assembly lines without needing any additional setup
- Maintenance and usability: The system is easy to maintain by Danfoss employees as it is built on their internal production IT interface. Any bugs or errors are easily fixed, and the system is put into running effectively immediately

www.manaraa.com

Functional Features

The dashboard application translates a given assembly line data into a visual representation to build predictive maintenance capabilities in the production facility. The application generates an analytic functionality after being trained with assembly line data. The system is capable of locating any possible discrepancies or negative trends in the data and therefore provides an interface with the existing Danfoss technology stack by extracting the data from the PLCs in real time. Besides, the machine can generate an alert in real time upon occurrence of a failure.

There are productive measures displayed on the dashboard. The production number is one attribute which is calculated which is a measure for the total number of units produced in a particular shift along with the difference between the expected value and actual value of the produce. Shift Productivity is a measure of the total units completed versus the total number of units planned. In turn, units planned is a measure of the number of workers in the shift in an assembly line and the time of the shift. Further, it gives the measure of the Overall Equipment Effectiveness (OEE) across the shift. Furthermore, the First Pass Yield (FPY) and the number of units which cleared the test stand are also shown along with the time remaining in a shift. The circle in the current Dashboard is already calculating FPY measure based on a fixed rate of 1 unit per 6 minutes.

www.manaraa.com

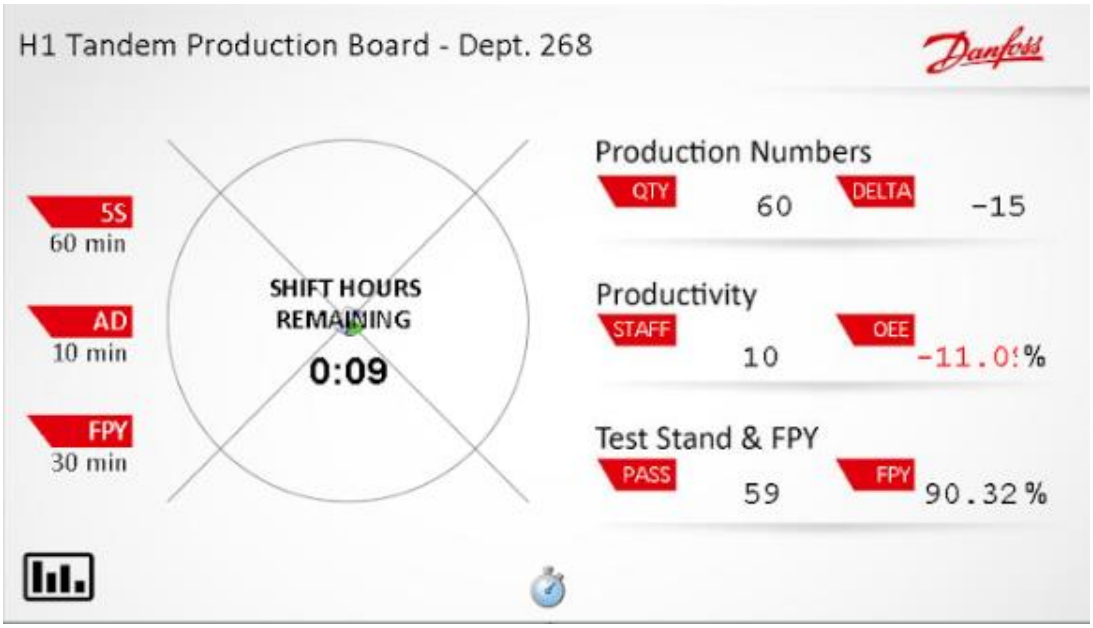

Figure 5: Ignition Interface showing the real-time dashboard system

The front-end look of the dashboard which is displayed on the screen on the floor has the circle replaced by a donut showing the overlap of the colors if the production is trailing or leading.

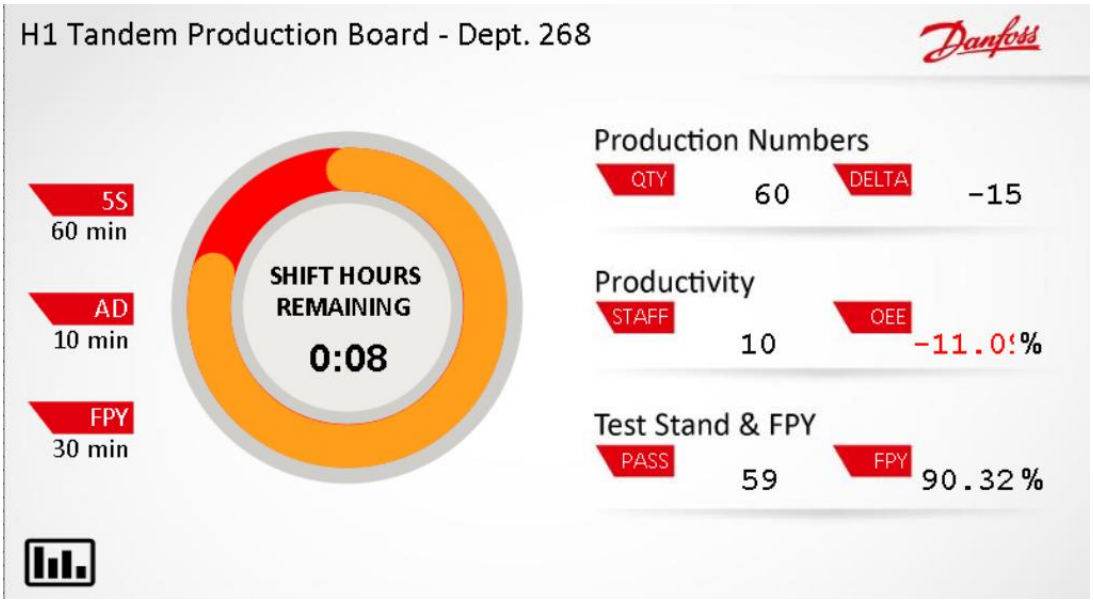

Figure 6: Final Front end display of the dashboard in the production facility

www.manaraa.com

The dashboard simultaneously demonstrates the wait times and the work times across the individual stations as well as the flow of the unit from one station to another. The average times across the stations throughout the shift is also stated across each station. The time represented in green is the work time and the time represented in red is the wait time. The horizontal line in yellow is a measure of the station having the highest efficiency or performance when compared to all the other stations.

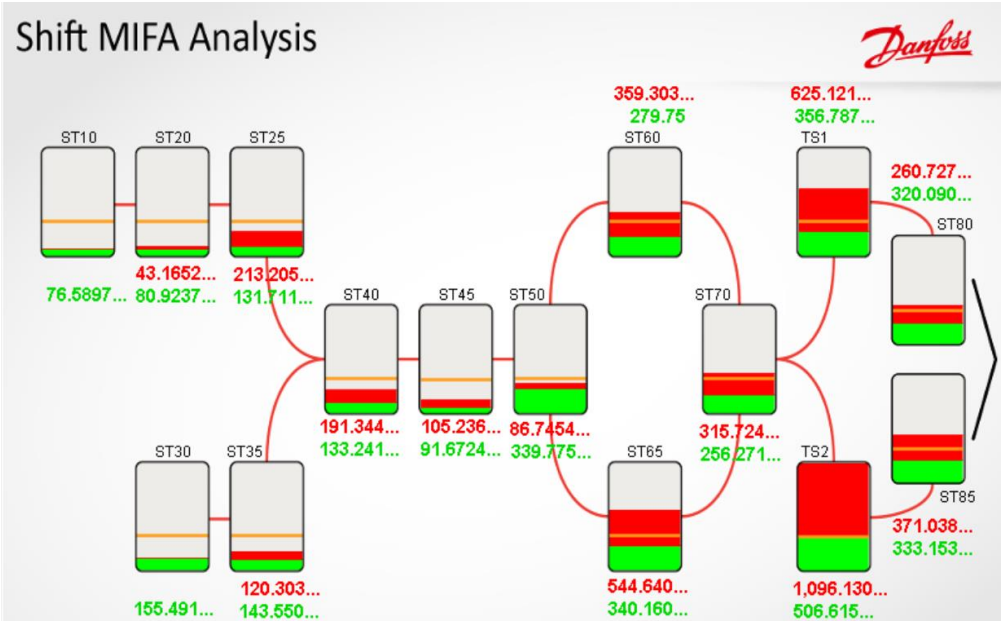

Figure 7: Shift Analysis across stations with work time and wait time

www.manaraa.com

It is essential to draw the attention to examine the trend of the work time or wait time across each station. Any notable difference in this trend assists in focusing at that particular station.

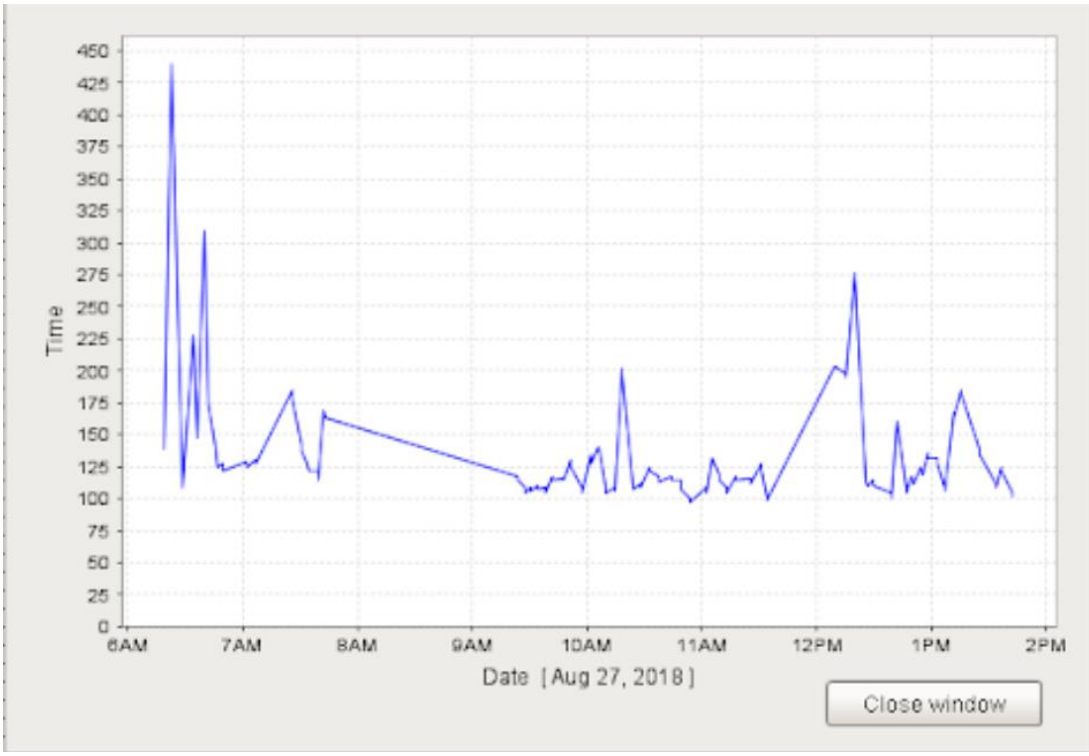

Figure 8: Work time of Station 10 during the first shift

www.manaraa.com

CHAPTER 7. CONCLUSION

The project is aimed at building an algorithm or a real-time monitoring system meant to increase the productivity of Danfoss' assembly lines. The system will monitor both the efficiency of the workers and the status of the machines on the line. This will send flags to the administrators should any of the allowed tolerances not be met. The user-interactive analytical framework allows the workers to monitor their own progress versus the expected progress for the day. The dashboard enables the continuous improvement team to determine the need and identify the scope for enhancing the business process within the production facility. The primary focus is to have the clean data to be the source for developing the dashboard for sending out prompts or alerts dynamically. Most of the historical data is cleaned by eliminating the vast considerable time difference of more than 7200 seconds between stations. Also, most of the missing data is replaced with zeros to reduce the complexity in data cleaning.

This dashboard when scaled across the entire organization, reduces the production loss due to downtime as it sends alert messages if a station takes longer than usual wait time. From a test stand-point of view, it supports the understanding of the frequency at which each product or unit is tested before the final step. Hence, the final solution when applied across other Danfoss assembly lines is expected to increase the productivity and efficiency and reduce runtime costs.

www.manaraa.com

REFERENCES

Wikipedia, Danfoss.com

Beebe, Ray S. Maintenance of Pumps using Condition Monitoring. Elsevier Advanced Technology; 2004.

Eisenmann RC. Machinery Malfunction Diagnosis and Correction: Vibration Analysis and Troubleshooting for the Process Industries. Prentice Hall; 1998.

Rao, B. K. N. Handbook of Condition Monitoring. Elsevier Advanced Technology; 1996.

Mohanty, S.; Jagadeesh, M.; Srivatsa, H. Big Data Imperatives: Enterprise Big Data Warehouse, BI Implementations and Analytics. Apress; 2013.

Predictive Maintenance, Position Paper - Deloitte Analytics Institute; 2017.

Perera Srinath: Alwis, Roshan; Machine Learning Techniques for Predictive Maintenance; 2017.

Computerweekly; Walker, Peter; UK country manager; Information Builders; 2011.

www.manaraa.com

APPENDIX

Data load Scripts on Ignition Server

```
# MIFA Analysis code for H1T line
# Created by R. Kornicki & Tejaswini Vuyyuru
# Last revised 6/27/2018

""" General variable definitions """
now = system.tag.read("[System]Gateway/CurrentDateTime").value

#now = system.date.addDays(now, -3)
currentDateTS = system.date.format(now, "yyyy-MM-dd") # date formatting for TS query
currentDateMES = system.date.format(now, "yyyy/MM/dd") # date formatting for MES query

print "hello"
shiftNo = 1

""" Query formulation for MES data pull
        Query returns station by station work start times and pivots those into an output table """

query = 'SELECT * '
query = query + '  FROM '
query = query + '( '
#-- This first CASE statement makes sure we always get the serial number, sometimes in
IN_SERIAL_NUMBER,
#-- other times it's the PALLET_SERIAL_NUMBER
query = query + 'SELECT CASE WHEN TRIM(ml.IN_SERIAL_NUMBER) IS NULL '
query = query + '          THEN ml.PALLET_SERIAL_NUMBER '
query = query + '          ELSE ml.IN_SERIAL_NUMBER'
query = query + '          END SERIAL_NUMBER '
query = query + '      ,mi.MODEL '
query = query + '      ,mi.SOURCE_ITEM || mi.ORDCODE MMC '
query = query + '      ,IN_LOCATION '
query = query + '      ,TSTAMP '
#-- This CASE statement ensures that we either get the pallet number of the subcomponent (front, rear,
center section)
#      -- OR BASE.  We need this since after the units are joined we get either NULL or the paint
pallet.
query = query + '      ,CASE WHEN TRIM(ml.IN_PALLET) IS NULL THEN \'BASE\' '
query = query + '           WHEN IN_LOCATION IN ( 145, 1145, 150, 1500, 165, 160, 1160, 170, 85,
255, 355, 180, 185, 1200, 460 ) THEN \'BASE\' '
query = query + '           ELSE LPAD(ml.IN_PALLET, 3, \'0\') '
query = query + '           END PALLET '
query = query + '  FROM MES_OWNER.MES_LOG ML '
query = query + '      INNER JOIN MES_OWNER.MES_WO_UNIT mwu ON mwu.SERIAL_NO =
ml.PALLET_SERIAL_NUMBER '
query = query + '      INNER JOIN MES_OWNER.MES_WOMAST mw ON mw.ID = mwu.WO_ID '
query = query + '     INNER JOIN MES_OWNER.MES_ITEM mi ON mi.ID = mw.ITEM_ID '

if shiftNo ==1:
        query = query + 'WHERE ml.TSTAMP >= TO_DATE(\" + currentDateMES + ' 06:00:00\',
\'YYYY/MM/DD HH24:MI:SS\') '
```

www.manaraa.com

```
                query = query + '   AND ml.TSTAMP <= TO_DATE(\" + currentDateMES + ' 13:59:59\',
\'YYYY/MM/DD HH24:MI:SS\') '

        if shiftNo ==2:
                query = query + 'WHERE ml.TSTAMP >= TO_DATE(\" + currentDateMES + ' 14:00:00\',
\'YYYY/MM/DD HH24:MI:SS\') '
                query = query + '   AND ml.TSTAMP <= TO_DATE(\" + currentDateMES + ' 21:59:59\',
\'YYYY/MM/DD HH24:MI:SS\') '

        if shiftNo ==3:
                startDate = system.date.format(system.date.addDays(now, -1), "yyyy/MM/dd")
                query = query + 'WHERE ml.TSTAMP >= TO_DATE(\" + startDate + ' 22:00:00\',
\'YYYY/MM/DD HH24:MI:SS\') '
                query = query + '   AND ml.TSTAMP <= TO_DATE(\" + currentDateMES + ' 05:59:59\',
\'YYYY/MM/DD HH24:MI:SS\') '

        query = query + '   AND mw.LINE_ID = 28 '
        query = query + '   AND (ml.IN_LOCATION < 461 OR ml.IN_LOCATION > 600) '
        #   -- This last condition excludes serial numbers that have ever had a DELETE SN called on them
        #   -- you might choose to ignore this if you are just looking at the last hour, you may not care
        query = query + '   AND 1 > ( '
        query = query + '             SELECT COUNT(*) '
        query = query + '               FROM MES_OWNER.MES_LOG ml2 '
        query = query + '              WHERE ml2.IN_SERIAL_NUMBER = ml.IN_SERIAL_NUMBER '
        query = query + '                AND ML2.PALLET_SERIAL_NUMBER =
ml.PALLET_SERIAL_NUMBER '
        query = query + '                AND CALLED_PROC IN (\'DELETE SN\') '
        query = query + '          ) '
        query = query + 'ORDER BY SERIAL_NUMBER, Pallet, TSTAMP '
        query = query + ') '
        query = query + 'PIVOT '
        query = query + '( '
        query = query + '    MIN (TSTAMP) '
        query = query + '    FOR IN_LOCATION '
        query = query + '     IN (10, 1110, 120, 1120, 125, 1125, 130, 1130, 135, 1135, 140, 1140, 145, 1145,
150, 1500, 160, 165, 1160, 170, 85, 255, 355, 180, 185, 1200, 460) '
        query = query + ') '
        query = query + 'ORDER BY SERIAL_NUMBER, PALLET '

        call = system.db.runQuery(query,"amsracp_dashboard")
        rawData = system.dataset.toDataSet(call)
        print call
        #reset call to 0 & release occupied memory
        #call = call[0]

        # re-formatting the returned table to include additional columns to be populated by Test Stand data
queries
        colCount = rawData.getColumnCount()
        columnDateData = []
        columnTS_ID = []
        columnWorkTime = []
        columnRouteTime = []

        for i in range(rawData.getRowCount()):
                columnDateData.append(now)
                columnTS_ID.append(0)
```

www.manaraa.com

```
		columnWorkTime.append(0.0)
		columnRouteTime.append(0.0)

	colPosition = rawData.getColumnIndex("85") # station 85 column index - two extra columns to be added after
	rawData = system.dataset.addColumn(rawData, colPosition + 1, columnTS_ID, "TS_ID", int)
	rawData = system.dataset.addColumn(rawData, colPosition + 2, columnDateData, "TestStart", type(now))
	#adding two columns at the end for product work time and route time comparisons
	rawData = system.dataset.addColumn(rawData, rawData.getColumnCount(), columnWorkTime, "WORK_TIME", float)
	rawData = system.dataset.addColumn(rawData, rawData.getColumnCount(), columnRouteTime, "ROUTE_TIME_DIFF", float)

	def getTestStartTime(serialNo, date):
		"""
		Outputs test start time of the Serial Number queried
		Args:
			SerialNo - [str] serial number of unit queried
			date     - [str] day queried YYYY-mm-dd
		Returns:
			[date] - test start date and time

		Internal data available:
			stand_id
			serial_number
			dt_created
			dt_eot
			passfail
			unit
		"""

		query = "select stand_id, serial_number, dt_created, dt_eot, passfail, unit "
		query = query + "from test_record where stand_id in (8,19) "
		query = query + "and dt_created>=to_date('" + date + "','YYYY-MM-DD') "
		query = query + "and serial_number='" + serialNo + "' "
		query = query + "and unit=1 ORDER BY DT_CREATED, UNIT"

		callts = system.db.runQuery(query,'Teststand')
		callts = system.dataset.toDataSet(callts)

		if callts.getRowCount() > 0:
			return [callts.getValueAt(0, "stand_id"), callts.getValueAt(0, "dt_created")]

		if callts.getRowCount() == 0:
			return   [0, 0]


	def averageTwoCol(dataset, col1, col2):
		"""
		Outputs the average difference in times of two columns within a PyDataSet
		Args:
			dataset (PyDataSet): dataset containing all timestamps
			col1 (str): name of first column of times
			col2 (str): name of second column of times
		Returns:
```

www.manaraa.com

```
                float: average seconds difference column 2 and column 1
        """
        if col1 == col2:
                return float(0)

        count = 0
        sum = 0
        for row in dataset:
                first = row[col1]
                second = row[col2]
                if first != None and second != None:
                        tdifference = system.date.secondsBetween(first, second)
                        if tdifference >= 0:
                                sum += tdifference
                                count += 1
        return sum / float(count)

def averageqBtw25n40(dataset, col1, col2):
        if col1==col2:
                return float(0)

        count = 0
        sum = 0
        for row in range(dataset.getRowCount()-3):
                if dataset.getValueAt(row, "SERIAL_NUMBER") == dataset.getValueAt(row+2,
"SERIAL_NUMBER"):
                        if dataset.getValueAt(row+3, "PALLET") == "BASE" or
dataset.getValueAt(row+2, "PALLET") == "BASE":
                                first = dataset.getValueAt(row, "1125")
                                second = dataset.getValueAt(row + 1, "140")
                                if first != None and second != None:
                                        tdifference = system.date.secondsBetween(first, second)
                                        if tdifference >= 0:
                                                sum += tdifference
                                                count += 1
        return sum / float(count)


def avgqBfw45(dataset, col1, col2):
        if col1==col2:
                return float(0)

        count = 0
        sum = 0
        for row in range(dataset.getRowCount()-3):
                if dataset.getValueAt(row, "SERIAL_NUMBER") == dataset.getValueAt(row+1,
"SERIAL_NUMBER"):
                        if dataset.getValueAt(row+3, "PALLET") == "BASE" or
dataset.getValueAt(row+2, "PALLET") == "BASE":
                                first = dataset.getValueAt(row+2, "1140")
                                second = dataset.getValueAt(row + 3, "145")
                                if first != None and second != None:
                                        tdifference = system.date.secondsBetween(first, second)
                                        if tdifference >= 0:
                                                sum += tdifference
                                                count += 1
```

www.manaraa.com

```
        return sum / float(count)

def tsq85(dataset,col1,col2):
        count = 0
        sum = 0
        for row in dataset:
                if row[25] != 0.0:
                        first = row[col1]
                        second = row[col2]
                        if first != None and second != None:
                                tdifference = system.date.secondsBetween(first, second)
                                if tdifference >= 0:
                                        sum += tdifference
                                        count += 1
        return sum / float(count)


def tsq1(dataset,col1,col2):
        count = 0
        sum = 0
        for row in dataset:
                if row[25] == 8.0:
                        first = row[col1]
                        second = row[col2]
                        if first != None and second != None:
                                tdifference = system.date.secondsBetween(first, second)
                                if tdifference >= 0:
                                        sum += tdifference
                                        count += 1
        return sum / float(count)

def tsq2(dataset,col1,col2):
        count = 0
        sum = 0
        for row in dataset:
                if row[25] == 19.0:
                        first = row[col1]
                        second = row[col2]
                        if first != None and second != None:
                                tdifference = system.date.secondsBetween(first, second)
                                if tdifference >= 0:
                                        sum += tdifference
                                        count += 1

        return sum / float(count)


# populating the Test Stand added columns with data from Test Stand Database
for a in range(rawData.getRowCount()):
        if rawData.getValueAt(a, "PALLET") == "BASE":
                serialNo = rawData.getValueAt(a, "SERIAL_NUMBER")
                tsquery = getTestStartTime(serialNo, currentDateTS)
                rawData = system.dataset.setValue(rawData, a, colPosition + 1, tsquery[0])
                rawData = system.dataset.setValue(rawData, a, colPosition + 2, tsquery[1])
```

www.manaraa.com

```
"""" Route times comparisons """"

controlRes = ["M1", "M2", "M3", "M4", "M5", "M6", "D9", "H4"]
shaftRes = ["B9", "C3", "F2", "F3", "F4", "E1"]
chargeRes = ["M"]

print rawData.getColumnNames()

funits = 0

if rawData.getRowCount() > 4:
        for row in range(rawData.getRowCount()-3):
                if rawData.getValueAt(row, "SERIAL_NUMBER") == rawData.getValueAt(row+3,
"SERIAL_NUMBER") and rawData.getValueAt(row+3, "PALLET") == "BASE":
                        if rawData.getValueAt(row, "10") != None and rawData.getValueAt(row+1,
"10") != None and rawData.getValueAt(row+3, "460") != None:

                                funits += 1

                                st10fW = st10rW = st20fW = st20rW = st25fW = st25rW =
st30cW = st35cW = st40frW = st40cW = st45bW = st50bW = st6065bW = st70bW = testTime = st8085bW = 0

                                if rawData.getValueAt(row, "10") != None and
rawData.getValueAt(row, "1110") != None:
                                        st10fW =
system.date.secondsBetween(rawData.getValueAt(row, "10"), rawData.getValueAt(row, "1110"))

                                if rawData.getValueAt(row+1, "10") != None and
rawData.getValueAt(row+1, "1110") != None:
                                        st10rW =
system.date.secondsBetween(rawData.getValueAt(row+1, "10"), rawData.getValueAt(row+1, "1110"))

                                if rawData.getValueAt(row, "120") != None and
rawData.getValueAt(row, "1120") != None:
                                        st20fW =
system.date.secondsBetween(rawData.getValueAt(row, "120"), rawData.getValueAt(row, "1120"))
                                if rawData.getValueAt(row+1, "120") != None and
rawData.getValueAt(row+1, "1120") != None:
                                        st20rW =
system.date.secondsBetween(rawData.getValueAt(row+1, "120"), rawData.getValueAt(row+1, "1120"))

                                if rawData.getValueAt(row, "125") != None and
rawData.getValueAt(row, "1125") != None:
                                        st25fW =
system.date.secondsBetween(rawData.getValueAt(row, "125"), rawData.getValueAt(row, "1125"))
                                if rawData.getValueAt(row+1, "125") != None and
rawData.getValueAt(row+1, "1125") != None:
                                        st25rW =
system.date.secondsBetween(rawData.getValueAt(row+1, "125"), rawData.getValueAt(row+1, "1125"))

                                if rawData.getValueAt(row+2, "130") != None and
rawData.getValueAt(row+2, "1130") != None:
                                        st30cW =
system.date.secondsBetween(rawData.getValueAt(row+2, "130"), rawData.getValueAt(row+2, "1130"))
```

www.manaraa.com

```
                                        if rawData.getValueAt(row+2, "135") != None and
rawData.getValueAt(row+2, "1135") != None:
                                                st35cW =
system.date.secondsBetween(rawData.getValueAt(row+2, "135"), rawData.getValueAt(row+2, "1135"))

                                        if rawData.getValueAt(row+1, "140") != None and
rawData.getValueAt(row+1, "1140") != None:
                                                st40frW =
system.date.secondsBetween(rawData.getValueAt(row+1, "140"), rawData.getValueAt(row+1, "1140"))
                                        if rawData.getValueAt(row+2, "140") != None and
rawData.getValueAt(row+2, "1140") != None:
                                                st40cW =
system.date.secondsBetween(rawData.getValueAt(row+2, "140"), rawData.getValueAt(row+2, "1140"))

                                        if rawData.getValueAt(row+3, "145") != None and
rawData.getValueAt(row+3, "1145") != None:
                                                st45bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "145"), rawData.getValueAt(row+3, "1145"))

                                        if rawData.getValueAt(row+3, "150") != None and
rawData.getValueAt(row+3, "1500") != None:
                                                st50bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "150"), rawData.getValueAt(row+3, "1500"))

                                        if rawData.getValueAt(row+3, "160") != None and
rawData.getValueAt(row+3, "1160") != None:
                                                st6065bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "160"), rawData.getValueAt(row+3, "1160"))

                                        if rawData.getValueAt(row+3, "165") != None and
rawData.getValueAt(row+3, "1160") != None:
                                                st6065bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "165"), rawData.getValueAt(row+3, "1160"))
                                        if rawData.getValueAt(row+3, "170") != None or
rawData.getValueAt(row+3, "85") != None:
                                                st70bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "170"), rawData.getValueAt(row+3, "85"))

                                        if rawData.getValueAt(row+3, "TestStart") != None or
rawData.getValueAt(row+3, "355") != None:
                                                testTime =
system.date.secondsBetween(rawData.getValueAt(row+3, "TestStart"), rawData.getValueAt(row+3, "355"))

                                        if rawData.getValueAt(row+3, "180") != None and
rawData.getValueAt(row+3, "1200") != None:
                                                st8085bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "180"), rawData.getValueAt(row+3, "1200"))
                                        if rawData.getValueAt(row+3, "185") != None and
rawData.getValueAt(row+3, "1200") != None:
                                                st8085bW =
system.date.secondsBetween(rawData.getValueAt(row+3, "185"), rawData.getValueAt(row+3, "1200"))

                                        unitWorkTime = (st10fW + st10rW + st20fW + st20rW +st25fW +
st25rW + st30cW + st35cW + st40frW + st40cW + st45bW + st50bW + st6065bW + st70bW + testTime +
st8085bW) / 3600.0
```

www.manaraa.com

```
                                        rawData = system.dataset.setValue(rawData, row + 3,
"WORK_TIME", unitWorkTime)

                                        routeTime = 0.95 # route time for base tandem model

                                        mmc = rawData.getValueAt(row, "MMC")

                                        if mmc[10:12] in controlRes:
                                                routeTime = routeTime + 0.2
                                                #print "control restriction option: " + mmc[10:12]

                                        if mmc[18:20] in shaftRes:
                                                routeTime = routeTime + 0.1
                                                #print "shaft restriction option: " +  mmc[19:21]

                                        if mmc[31:32] in chargeRes:
                                                routeTime = routeTime + 0.25
                                                #print "charge pump restriction option: " + mmc[33:34]

                                        rawData = system.dataset.setValue(rawData, row +3,
"ROUTE_TIME_DIFF", unitWorkTime - routeTime)
                                        #print " - acc. time of " + str("%.2f" % unitWorkTime) +" hours
with difference of " + str("%.2f" % (unitWorkTime - routeTime))

        print rawData



        def stationWorkTime(dataset, station):

                headers = ["Time", "Work Time"]
                data = []
                for row in range(dataset.getRowCount()):
                        if dataset.getValueAt(row, station) != None and dataset.getValueAt(row,
dataset.getColumnIndex(station)+1) != None:
                                tdifference = system.date.secondsBetween(dataset.getValueAt(row,
station),  dataset.getValueAt(row, dataset.getColumnIndex(station)+1))
                                data.append([dataset.getValueAt(row, station), tdifference])

                graphData = system.dataset.sort(system.dataset.toDataSet(headers, data), 0)

                return graphData


        #system.tag.write("graph data", stationWorkTime(rawData, "10"))
        rawData2 = system.dataset.toPyDataSet(rawData)

        # Calls functions for each station
        st10w = averageTwoCol(rawData2,4,5)
        st20q = averageTwoCol(rawData2,5,6)
        st20w = averageTwoCol(rawData2,6,7)
        st25q = averageTwoCol(rawData2,7,8)
        st25w = averageTwoCol(rawData2,8,9)
        # Looks for "BASE" and goes to st40 that jumps rows
        st40q1= averageqBtw25n40(rawData,9,14)
        st30w = averageTwoCol(rawData2,10,11)
```

www.manaraa.com

```
st35q = averageTwoCol(rawData2,11,12)
st35w = averageTwoCol(rawData2,12,13)
st40q2= averageTwoCol(rawData2,13,14)
st40w = averageTwoCol(rawData2,14,15)
# Speacial procedure that jumps one row down
st45q = avgqBfw45(rawData,15,16)
st45w = averageTwoCol(rawData2,16,17)
st50q = averageTwoCol(rawData2,17,18)
st50w = averageTwoCol(rawData2,18,19)
st60q = averageTwoCol(rawData2,19,20)
st65q = averageTwoCol(rawData2,19,21)
st60w = averageTwoCol(rawData2,20,22)
st65w = averageTwoCol(rawData2,21,22)
st70q = averageTwoCol(rawData2,22,23)
st70w = averageTwoCol(rawData2,23,24)
st85q = tsq85(rawData2,24,26)
ts1q  = tsq1(rawData2,24,26)
ts2q  = tsq2(rawData2,24,26)
ts1w  = tsq1(rawData2,26,28)
ts2w  = tsq2(rawData2,26,28)
st180q= averageTwoCol(rawData2,28,29)
st185q= averageTwoCol(rawData2,28,30)
st180w= averageTwoCol(rawData2,29,31)
st185w= averageTwoCol(rawData2,30,31)
st1200= averageTwoCol(rawData2,31,32)


#Required Dataset
headers = ["st10", "st20", "st25", "st40q1", "st30", "st35", "st40", "st45", "st50", "st60", "st65", "st70",
"st85", "ts1", "ts2", "st180", "st185", "st1200"]
averageTimes = []
##Appending work times
averageTimes.append([st10w, st20w, st25w, 0.0, st30w, st35w, st40w, st45w, st50w, st60w, st65w,
st70w, 0.0, ts1w, ts2w, st180w, st185w, st1200])

##Appending queue times
averageTimes.append([0.0, st20q, st25q, st40q1, 0.0, st35q, st40q2, st45q, st50q, st60q, st65q, st70q,
st85q, ts1q, ts2q, st180q, st185q, 0.0])
averageTimesData = system.dataset.toDataSet(headers, averageTimes)

system.tag.write("ShiftCounts/mifaData", averageTimesData)
system.tag.write("ShiftCounts/stationsData", rawData)
system.tag.write("ShiftCounts/st10w", st10w)
system.tag.write("ShiftCounts/st20w", st20w)
system.tag.write("ShiftCounts/st20q", st20q)
system.tag.write("ShiftCounts/st25w", st25w)
system.tag.write("ShiftCounts/st25q", st25q)
system.tag.write("ShiftCounts/st30w", st30w)
system.tag.write("ShiftCounts/st35w", st35w)
system.tag.write("ShiftCounts/st35q", st35q)
system.tag.write("ShiftCounts/st40w", st40w)
system.tag.write("ShiftCounts/st40q2", st40q2)
system.tag.write("ShiftCounts/st45w", st45w)
system.tag.write("ShiftCounts/st45q", st45q)
system.tag.write("ShiftCounts/st50w", st50w)
system.tag.write("ShiftCounts/st50q", st50q)
```

www.manaraa.com

```
system.tag.write("ShiftCounts/st60w", st60w)
system.tag.write("ShiftCounts/st60q", st60q)
system.tag.write("ShiftCounts/st65w", st65w)
system.tag.write("ShiftCounts/st65q", st65q)
system.tag.write("ShiftCounts/st70w", st70w)
system.tag.write("ShiftCounts/st70q", st70q)
system.tag.write("ShiftCounts/ts1w", ts1w)
system.tag.write("ShiftCounts/ts1q", ts1q)
system.tag.write("ShiftCounts/ts2w", ts2w)
system.tag.write("ShiftCounts/ts2q", ts2q)
system.tag.write("ShiftCounts/st180w", st180w)
system.tag.write("ShiftCounts/st180q", st180q)
system.tag.write("ShiftCounts/st185w", st185w)
system.tag.write("ShiftCounts/st185q", st185q)

window = system.nav.openWindow('mifa board')
system.nav.centerWindow(window)
```

Tag Scripts on Ignition Server

```
# This script was generated automatically by the navigation
# script builder. You may modify this script, but if you do,
# you will not be able to use the navigation builder to update
# this script without overwriting your changes.


window = system.nav.openWindow('Station plot', {'stationName' : 10})
system.nav.centerWindow(window)


dataset = system.tag.read("ShiftCounts/stationsData").value

stationName = "TestStart"

headers = ["Time", "Work Time"]
data = []
for row in range(dataset.getRowCount()):
        if dataset.getValueAt(row, dataset.getColumnIndex(stationName)-1) == 8.0:
                if dataset.getValueAt(row, stationName) != None and dataset.getValueAt(row,
dataset.getColumnIndex(stationName)+2) != None:
                        tdifference = system.date.secondsBetween(dataset.getValueAt(row,
stationName),  dataset.getValueAt(row, dataset.getColumnIndex(stationName)+2))
                        data.append([dataset.getValueAt(row, stationName), tdifference])
                        graphData = system.dataset.sort(system.dataset.toDataSet(headers, data), 0)


system.tag.write("ShiftCounts/stationCycleTimePlot", graphData)
```

Tag Scripts for non-test stand stations on Ignition Server

```
# This script was generated automatically by the navigation
# script builder. You may modify this script, but if you do,
# you will not be able to use the navigation builder to update
```

www.manaraa.com

```
	# this script without overwriting your changes.


	window = system.nav.openWindow('Station plot', {'stationName' : 10})
	system.nav.centerWindow(window)


	dataset = system.tag.read("ShiftCounts/stationsData").value

	stationName = "10"

	headers = ["Time", "Work Time"]
	data = []
	for row in range(dataset.getRowCount()):
		if dataset.getValueAt(row, stationName) != None and dataset.getValueAt(row,
dataset.getColumnIndex(stationName)+1) != None:
			tdifference = system.date.secondsBetween(dataset.getValueAt(row, stationName),
dataset.getValueAt(row, dataset.getColumnIndex(stationName)+1))
			data.append([dataset.getValueAt(row, stationName), tdifference])

			graphData = system.dataset.sort(system.dataset.toDataSet(headers, data), 0)

	system.tag.write("ShiftCounts/stationCycleTimePlot", graphData)
```

www.manaraa.com