Danfoss Predictive Maintenance

FINAL REPORT

Team 32
Client: Danfoss
Adviser: Namrata Vaswani
Derek Bruun - Embedded Systems Lead / Hardware Integrations
Micky Lindsay - Software Integrations Lead / Dev Tools Manager
Smriti Manral - Lead Test Engineer / Test Designer
Victoria Rasavanh - Communications Lead / Webmaster
Jess Walters - Lead Architect / Tech Lead
http://sdmay18-32.sd.ece.iastate.edu/

Table of Contents

1 Introduction

1.1 Project statement

According to their official website, “Danfoss Group is a global producer of products and services used in areas such as cooling food, air conditioning, heating buildings, controlling electric motors, compressors, bowling, drives and powering mobile machinery”. Today, Danfoss has several machine lines that are responsible for the production of these goods. Each machine line has various stations which are handled directly by assembly line workers. Each worker is incharge of processing a product and transferring it from one station to another until it reaches the last station in the line. The time elapsed between the first station and last station is used for measuring the overall productivity of a line. Being able to measure productivity and keep record of production cycles is essential in running their business successfully.

Currently, most of Danfoss’s assembly line metrics are stored in a non-centralized way. Analysis of this data is often done manually, costing time and effort spent making critical business decisions. Because of this, Danfoss has requested our team (sdmay18-32) create a solution that maximizes the use of the gathered data, and generates useful metrics in real-time. To do this, the team has designed and prototyped a dashboard that visually streamlines the data into usable metrics, such as daily production count or first pass yield (FYP). Data is sourced from sensors attached to assembly line workstations, and databases associating workstations to various assembly lines.

After creating the visualization, an extended goal would be to create forecast metrics using machine learning. Such a forecast would allow Danfoss to predict maintenance scheduling for machines, or anticipate the number of worker output needed to meet certain production goals.

1.2 Purpose

Today, a large number of modern manufacturing companies are starting to incorporate data analytics and machine learning in their production and manufacturing process. This kind of data analytics can help in collecting metrics and computing relevant numbers essential in measuring and improving production process. Knowing the number of items produced by each worker during their shifts can help the company keep track of performance of each worker and see how they can improve it. Basically this specific metric data will help them manage their workers more efficiently. Metrics showing the quantity of products produced daily and during the shift can help the company see if they are missing something or doing something wrong and based on those numbers help figure out how they can improve the number of good being produced daily or during the shift.

Predicting the system underperformance with machine learning models can help improve the manufacturing process by possibly telling the company before in hand how the future production numbers are going to look and whether to expect system under performance or over performance, based on which the company can decide ahead of time what they could do to improve the production process incase if system underperformance is detected by the model.

1.3 Goals

1. Solution Goals
    a. Analyze the data given to us by Danfoss and understand the relationship between various columns

b. Design a dashboard that displays useful metrics based on the data analytics done
c. Design a machine learning model to predict system underperformance

2. Senior Design Team Goals
    a. Everyone should contribute an equal amount of work to the end product
    b. If scheduling conflicts arise, the team should be notified in advance

2 Deliverables

2.1 Client Dashboard

The client dashboard is a visual representation of an assembly line's overall productivity. Data includes, but is not necessarily limited to, first pass yield, time passed in the shift, units produced this shift, total number of tests passed, overall equipment effectiveness, and overall process effectiveness. See Section 3: Design and Appendix B: Software Requirements for details.

2.1 Source code

All code will be version controlled for easy contribution and management. By the project's end, the repository will be exported and made available to Danfoss personnel. The majority of this code is Python.

2.3 documentation

Documentation should cover both technical and non-technical aspects of the project. This includes the original Project Plan and Design document created for the senior design class (S E 491 and S E 492), and this final report. Technical documentation includes instructions for setup, execution, maintenance, and design of the solution itself, while non-technical documentation will include the project's goals and background.

3 Design

3.1 System specifications

The final solution is to be displayed on monitors that oversee that assembly line floors. Both assembly line workers and managers should be able to see the dashboard's content from afar. The design should not be so distracting as to reduce productivity or put any workers in a position of danger. Managers should be able to use the dashboard to make informed decisions regarding the productivity of their team, and the workers themselves should be able to reference the dashboard to gauge their own productivity levels.

Behind the scenes, the system also needs to be managed by Danfoss developers. Their system is largely treated as a black box for the purposes of the project, but will be represented by a MySQL database that interfaces with Ignition to display data.

Please refer Appendix B for the full list of requirements for this project

3.2 Proposed Design

Regarding the dashboard deliverable, the team will utilize the existing Danfoss tech stack, namely Industrial Automation's Ignition. This solution provides the basis of the UI and display functionality, while our team will generate the data to display. Data will be gathered and sanitized from the existing MES database structure wherein the client stores all relevant product line statistics.

The data analysis portion of the project will utilize this same data, and will analyze the data finding statistics such as, first pass yield, and work time. Once the data is analyzed, it is piped into the Ignition dashboard to be displayed on the assembly line.

3.3 Design Analysis

The final design meets all of our functional and non-functional requirements. The final solution is based on Danfoss' network stack. The system is designed to integrate fully with the MES databases as well as the Ignition frontends deployed to the assembly lines.

4 Testing/Development

4.1 Client Dashboard Specifications

A primary component of our solution is a client dashboard that faces the workers on the assembly line floor. Notifications should be provided to management users regarding overall production or previous production cycles. The left hand side of the visual consists of a high-level, circular representation of the shift's overall status. The right hand side contains some other numeric data such as first pass yield (FYP), overall equipment effectiveness (OEE), and and overall process effectiveness (OPE). A second interface has been proposed that shows the timestamps of specific stations, their production output, and their relationship with other workstations in the line.

4.2 Hardware/software

The software we are using to display our dashboard is Ignition, a software that uses Python as its primary language to communicate with the server side and display the processed information to the workers on the assembly line. The data is gathered from light walls and PLCs on the assembly lines. Light wall information is sent to the database for later processing while Ignition is able to hook directly into the PLCs while still storing this information in the database for later access.

4.3 Process

Code: Testing & Development

- *Input Verification* - verification has been done to ensure no duplicates are put into the system as well as no false data is intentionally entered.
- *Display Verification* - verification has been done to ensure that the data we are pulling to be process is both correct and formatted in the expected way.
- *Unit Tests* - unit tests are planned to ensure our efficiency values, first pass yield and other calculated values are correctly calculated and displayed.

- *Local Setup* - as a raw, exported Ignition project is difficult to source control, so code for individual scripts are stored to GitLab, and manually imported to an Ignition project.
- *Feedback* - to ensure an effective visualization, discussions on usability and content occurred regularly

Testing the System

- *First Pass Yield* - Verify that number of retested parts
- *Units Produced* - Verify workers are meeting production quotas more often
- *Usability* - Receive feedback from workers on intuitiveness of UI.
- *Load Tests* - Perform load testing by creating production simulation, to ensure the system is performing as intended under normal conditions.

5 Results & Implementation Details

Our final deliverable for this project was our dashboard that would be facing the worker end. This dashboard uses information gathered from the assembly line to keep the workers up-to-date on how their progression through the day is compared to the expected values. This dashboard is initially only running on a single line but can be easily scaled up to other lines by simply creating a copy of the dashboard and pointing it to the correct PLCs on the new line. For a more detailed description of how to set up and use the system refer to Appendix I. Unfortunately due to time constraints and changes of scope, we have been unable to reach all testing goals.

6 Conclusions

This project has been a big lesson in how a project's scope may change over time; we went from prototyping a robust, machine learning-based system to prototyping a system geared toward proper data management and visualization. Furthermore, a recent scope change adds yet another feature for a future senior design team to work on, entailing another data display. Given our time constraints, implementation of testing and this new, secondary dashboard would be tasks for the Danfoss team and/or any other senior design groups that work on this project. Ultimately, had the scope been set in stone much earlier, it is likely that more concise development work may have gotten done.

7 References

[1] Olsson, E., Funk, P. and Xiong, N. (2004). Fault Diagnosis in Industry Using Sensor Readings and Case-Based Reasoning, *Journal of Intelligent & Fuzzy Systems*, 15, (41- 46).

8 Appendices

Appendix A: Table of Incurred Costs

Item	Cost/Unit	Quantity	Incurred Cost
MSI GTX 1080 TI AERO 11G OC Video Graphic Card	$824	2	$1648
Intel i7-7700k	$330	1	$330
64gb Corsair DDR4 ram	$740	1	$740
Asus Z270-WS motherboard	$379	1	$379
512Gb Samsung 960 pro m.2 NVMe SSD	$289	1	$289
Corsair HX1000 PSU	$199	1	$199
Thermaltake X5 TG	$159	1	$159
Ignition Software License (6-month)	Free	4	$0
MySQL Workbench	Free	1	$0
SQL Server Management Studio	Free	1	$0
SQL Server Express	Free	1	$0
		Total Cost	$3744

Appendix B: Software Requirements

Definitions

- *Users* - users of the system include assembly line workers, assembly line management, and developers
  - *Workers* - refer to employees who interact with the assembly line stations themselves.
  - *Management* - refer to employees who must make informed decisions based on data shown by the dashboard.
  - *Developers* - refer to employees charged with maintenance and feature management.
- *Client Dashboard* - may simply be referred to as "the dashboard". Interprets data from the database to be shown visually.
- *Ignition Server* - represents the live datastore of the system. Mainly tasks with making requests from the database, and updating in real-time.
- *Database* - For the purposes of the project, the database is stored offsite on Iowa State University's campus. When integrated with Danfoss architecture, it will most likely be stored on site rather than in the cloud.
- *Shift* - a parameterized time frame representing hours that workers are able to work within.
  - Standard Shifts: 6AM-2PM, 2PM-10PM, and 10PM-6AM
  - Special Shifts: 6:15AM-7AM,7AM-8AM, and 8:15AM-9AM

Non-Functional Requirements

Our non-functional requirements define non-technical aspects of our project. Such aspects include:

1. *Finance* - The project must stay within the allocated $8, 000 budget (see Appendix A)
2. *Performance* - The system should be able to update quickly and respond to updates immediately.
   a. The dashboard should maintain responsiveness with minimal amounts of data provided
   b. The dashboard should maintain responsiveness with large amounts of data provided
   c. The dashboard should be able to load the data in a timely manner when an assembly line switch occurs
   d. The dashboard should be able to load the data in a timely manner when a shift switch occurs
3. *Security* - As we will be "mirroring" the production environment with our own, isolated database populated with Danfoss data, care must be taken to ensure our interactions with it does not expose classified information.
   a. The database must be accessed using a VPN
   b. The database must only consist of a subset of production data
4. *Stability* - The system should be stable and bug-free. We need maximum uptime to continue running analytics on the assembly line.
   a. The system should be resistant if the inflow of data stops
   b. The system should be resistant to sudden, extreme increases of data inflow/outflow
5. *Scalability* - For the scope of the project, we will test and design our system based off the data of one assembly line. However, it is very possible that the system will be applied to multiple assembly lines that will be piping data into the system for long periods of time.
   a. The dashboard must adaptable to multiple, and/or differently structured, assembly lines.
   b. The dashboard must be able to handle data inflow/outflow for an entire assembly line shift
   c. The dashboard must be able to handle data inflow/outflow for multiple assembly line shifts
   d. The dashboard must be able to switch from one assembly line to another

6. *Maintainability*
    a. Developers should be able to maintain or change the project remotely
    b. Development should be able to get done collaboratively
    c. The dashboard must be created through Ignition
    d. Python must be employed as the primary coding/scripting language
7. *Usability*
    a. Workers must be able to deduce what each visual component represents.
    b. Workers should *not* be able to change parameters regarding ideal productivity goals
    c. Management must be able to deduce what each visual component represents
    d. Management should be able to change parameters regarding ideal productivity goals easily in response to changing business needs
8. *Documentation* - Information to be provided on the system as a whole
    a. Documentation must be provided regarding setup, execution, and maintenance
    b. Documentation must provided regarding extendability

Functional Requirements

The created system has the following functional requirements:

1. *Client* - tasked with displaying the data, and updating in real-time
    a. *Green Status* - represent the current assembly line performance as optimal or ideal
    b. *Orange Status* - represents the current assembly line performance as acceptable, slightly nonoptimal, and/or at risk of becoming non-ideal.
    c. *Red Status* - color changes to red indicate significant underperformance, or non-ideal performance
    d. *Display Single Shift* - on one shift's dataset may be displayed at a time
    e. *Display Line Name* - the line name must be displayed above the data
    f. *Display Time* - the time passed within a shift must be displayed, and must be able to turn over once a new shift starts.
    g. *Display Total Units* - the total number of units produced in this line during this particular shift must be displayed in comparison
    h. *Display First Pass Yield* - the percent of tests that passed on the first time must be displayed for its given shift.
    i. *Display Overall Equipment Efficiency* - represents the average efficiency of each individual station of the assembly line as a percent. Formulated as work time of an individual station divided by the sum of work time and idle time for an individual station.
    j. *Display Overall Process Efficiency* - represents the average efficiency of each individual station of the assembly line as a percent. Formulated as the total work time of the entire line divided by the sum of the total work time and total idle time of the entire line.
2. *Server* - tasked with fetching data from the database, and sending data to the client
    a. *Database Requests* - the Ignition server must be able to make requests from the database.
    b. *Data Parsing* - the server must trim and format the data to be readable by the client
    c. *Send Data to Client* - the server must be able to send parsed data to the client
3. *Database*
    a. The database must be able have data added to or extracted from it.
    b. The project database must only hold a subset of the production data
    c. The database must be able to simulate the production environment for testing.

Appendix I: Operations Manual

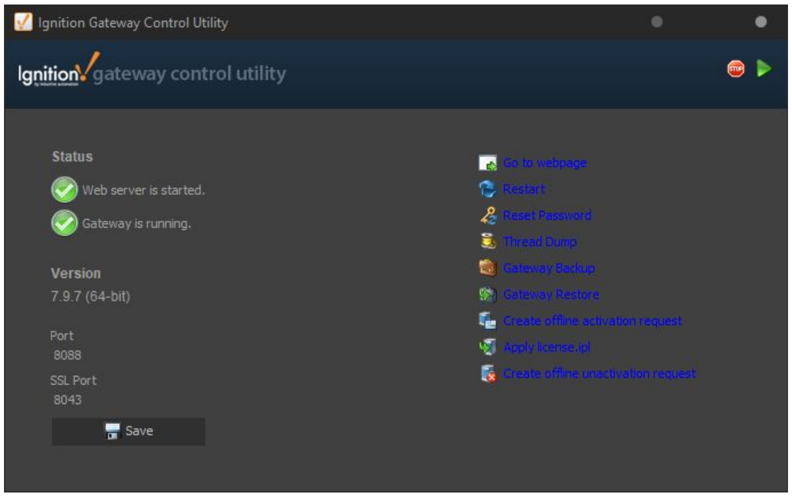
The proprietary Inductive Automation platform Ignition, used for dashboard display, is a software tool purchased and hosted by Inductive Automation clients. This server platform allows for communication between manufacturing companies' PLC (programmable logic controller) systems.

As the dashboards used for Danfoss require their existing layout of assembly lines and datastores, recreation of the platform is not feasible. Regardless, this appendix covers basic Ignition Server setup.

Licensing and download information available at https://inductiveautomation.com/Ignition/.

- Installation and Setup
  Through the link above, input the information (name, company, and email address) for the user and select the desired platform to run the Ignition Web Server and Gateway. Execute the provided binary from Inductive Automation, and proceed through the installation wizard.

  After installation and upon execution of the Ignition platform, the user is presented with the following control utility.

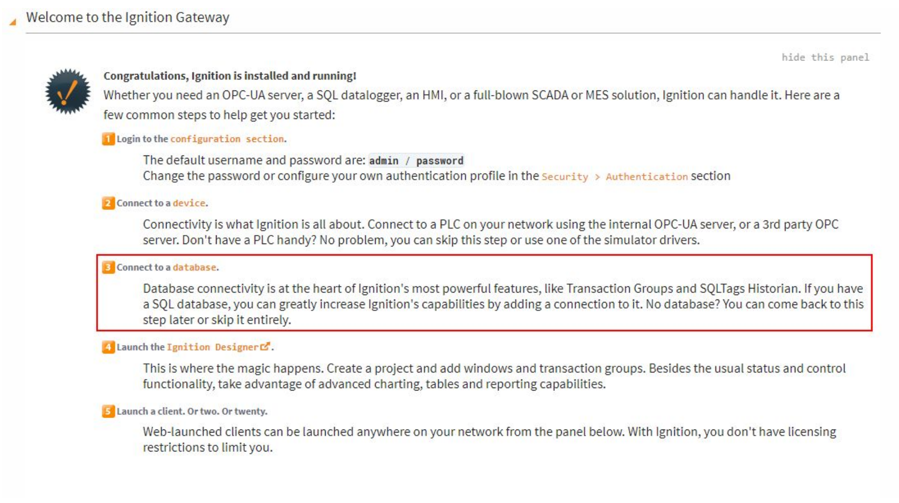

Ignition Gateway Control Utility
Ignition gateway control utility
Go to webpage
Restart
Reset Password
Thread Dump
Gateway Backup
Gateway Restore
Create offline activation request
Apply license.ipl
Create offline unactivation request
Save

Status
Web server is started.
Gateway is running.
Version
7.9.7 (64-bit)
Port
8088
SSL Port
8043

  By default the web server is hosted at the default port "`8088`". When accessed through "`http://localhost:8088`", the user is presented with the Ignition Gateway homepage. This page is used to configure the server, launch the Ignition designer, and execute dashboards. Initially logged out, the user must select the login option and input the default credentials of "`admin / password`" for username and password respectively.

  It is important to note, that the free license of Ignition will only run for two hours before needing to be reset through the Gateway.

- Database Connection
  Although the user has the ability to connect an existing PLC system, for the purposes of this project the only connection which needs to be made is with an existing relational database. This can be done after login from the Gateway homepage through the third available option titled "`Connect to a database`" highlighted below in red.


Welcome to the Ignition Gateway

hide this panel

Congratulations, Ignition is installed and running!
Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get you started:

1 Login to the `configuration section`.

The default username and password are: `admin / password`
Change the password or configure your own authentication profile in the `Security > Authentication` section

2 Connect to a `device`.

Connectivity is what Ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? No problem, you can skip this step or use one of the simulator drivers.

3 Connect to a `database`.

Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can greatly increase Ignition's capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.

4 Launch the `Ignition Designer`.

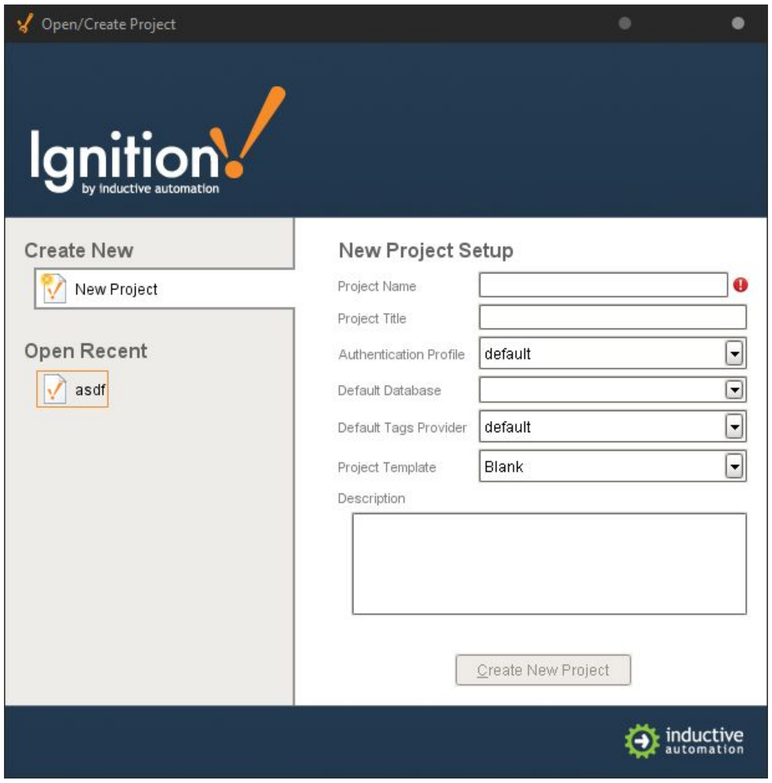
This is where the magic happens. Create a project and add windows and transaction groups. Besides the usual status and control functionality, take advantage of advanced charting, tables and reporting capabilities.

5 Launch a client. Or two. Or twenty.

Web-launched clients can be launched anywhere on your network from the panel below. With Ignition, you don't have licensing restrictions to limit you.

Ignition supports an number of database connections including MySQL, Oracle, Firebird, etc. The user may choose whichever database system they like. The execution of this project, without manual modification, requires the specific database schema listed within the Appendix below. This schema is required, due to the various database queries within the project. Such queries which retrieve specific serial numbers within specific stations on the existing Danfoss assembly lines require this.

- Project Creation and Import
  To access dashboard creation, import, and modification, the user must execute the Ignition Designer from the "`Launch Designer`" button at the upper right of the Gateway homepage. This link will download a java executable "`designer.jnlp`" which opens the main Ignition design tool. This tool will require the user to authenticate again with the same admin credentials.



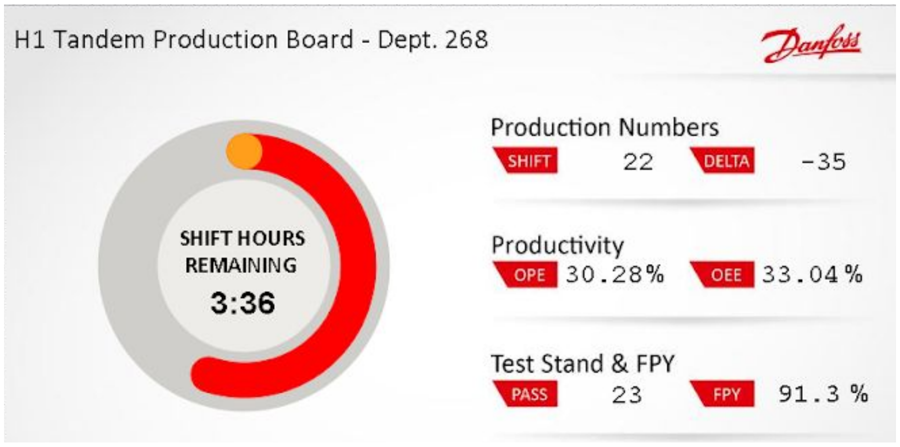
New Project Setup

Project Name	
Project Title	
Authentication Profile	default
Default Database	
Default Tags Provider	default
Project Template	Blank
Description	

Upon running the designer and logging in, the user is greeted with a project creation screen. A blank project is required before this document's project can be imported. The user must also select the previously created database connection which houses the correct schema.

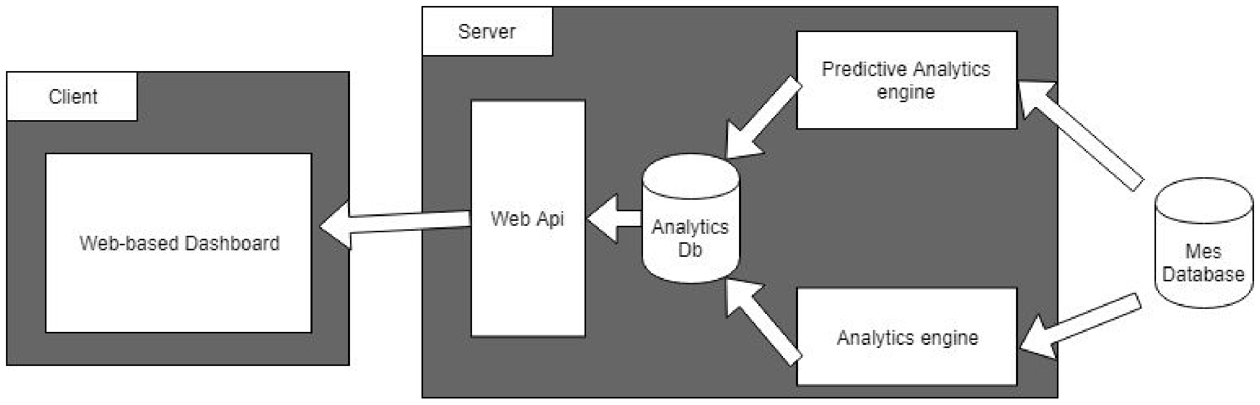
After creation, the project (.proj file) can be imported into this blank Ignition project. To do so, the user must access the Import dialog through the `File>Import` menu. Select the desired project file and complete the import. At this point, the project can be saved and published through the same file menu dropdown.

- Dashboard Execution
  Now that the project and dashboard is published, the user may return to the Gateway browser page and navigate to the homepage. The user will now see the named project under the Launch Project panel. Once the user has selected the project and clicked the Launch button, another jnlp file will be downloaded and can be executed. After login, and if the Database connection is complete with the correct schema and data entry, the following dashboard display will be visible.



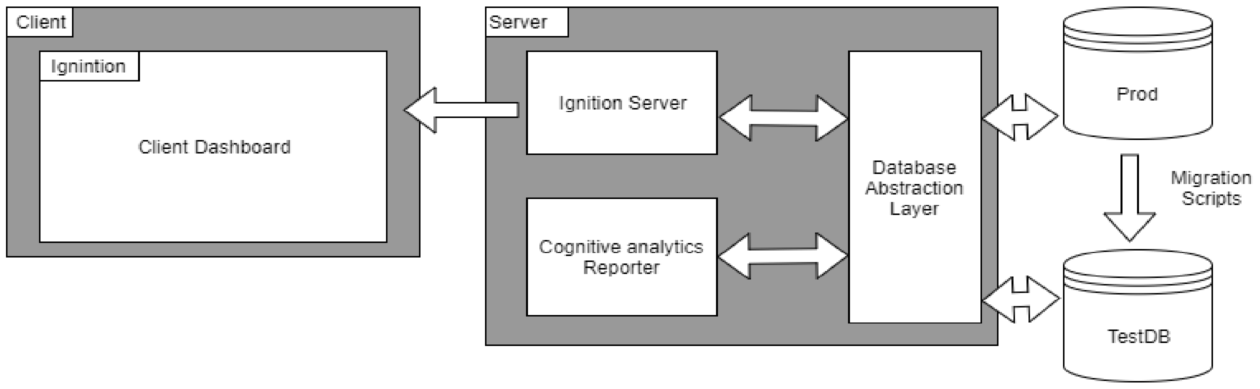
Appendix II: Alternative Solutions and Srchitectures

Initial Proposed Architecture



The initial proposed architecture revolved around the use of a web-based client connected to a server side api. The api would grab data from a database containing data analyzed by the analytics engines. This architecture was planned before the requirement to use Ignition was added.

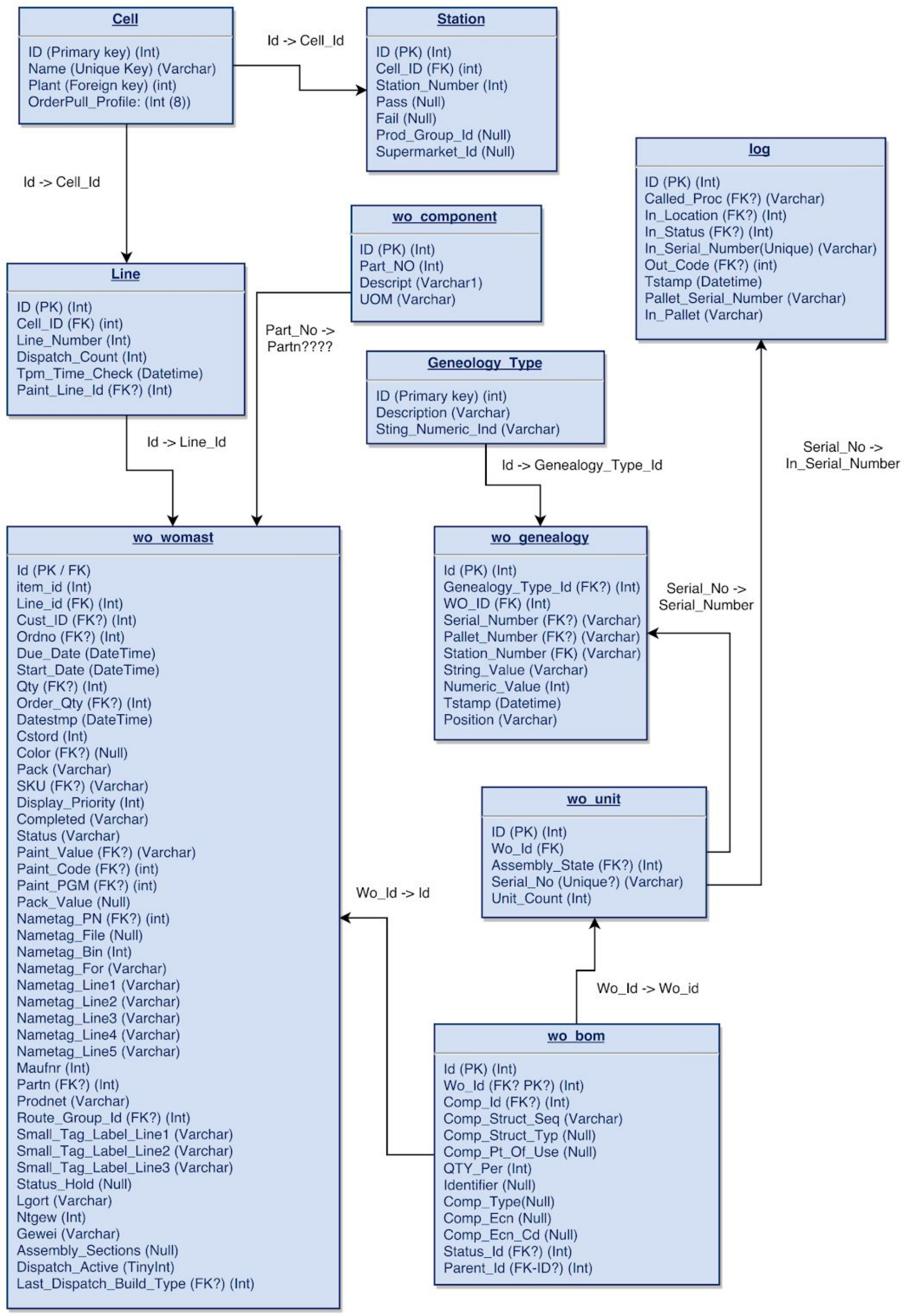
Proposed Ignition-Based Architecture



After the Ignition requirement was added the project architecture was reimagined to utilize the Ignition stack including the client and data abstraction layer allowing us to access multiple different database types. This proved to be very convenient since both our production database and test database were of different languages.

MES Test Database

Below is the schema for our test database. When we were initially given a large amount of MES data in the form of a multiple CSV files. We had to reconstruct the data schema and translate it into a format that the team was familiar with. We reconstructed the tables and types with a custom made Python script to migrate the data into a new schema. The relationships were then determined and the primary and foreign key pairs were added by hand using MySQLWorkbench.


Cell
ID (Primary key) (Int)
Name (Unique Key) (Varchar)
Plant (Foreign key) (int)
OrderPull_Profile: (Int (8))
Id -> Cell_Id
Station
ID (PK) (Int)
Cell_ID (FK) (int)
Station_Number (Int)
Pass (Null)
Fail (Null)
Prod_Group_Id (Null)
Supermarket_Id (Null)
log
ID (PK) (Int)
Called_Proc (FK?) (Varchar)
In_Location (FK?) (Int)
In_Status (FK?) (Int)
In_Serial_Number(Unique) (Varchar)
Out_Code (FK?) (int)
Tstamp (Datetime)
Pallet_Serial_Number (Varchar)
In_Pallet (Varchar)
Id -> Cell_Id
wo_component
ID (PK) (Int)
Part_NO (Int)
Descript (Varchar1)
UOM (Varchar)
Line
ID (PK) (Int)
Cell_ID (FK) (int)
Line_Number (Int)
Dispatch_Count (Int)
Tpm_Time_Check (Datetime)
Paint_Line_Id (FK?) (Int)
Part_No -> Partn????
Geneology_Type
ID (Primary key) (int)
Description (Varchar)
Sting_Numeric_Ind (Varchar)
Id -> Line_Id
Id -> Genealogy_Type_Id
Serial_No -> In_Serial_Number
wo_womast
Id (PK / FK)
item_id (Int)
Line_id (FK) (Int)
Cust_ID (FK?) (Int)
Ordno (FK?) (Int)
Due_Date (DateTime)
Start_Date (DateTime)
Qty (FK?) (Int)
Order_Qty (FK?) (Int)
Datestmp (DateTime)
Cstord (Int)
Color (FK?) (Null)
Pack (Varchar)
SKU (FK?) (Varchar)
Display_Priority (Int)
Completed (Varchar)
Status (Varchar)
Paint_Value (FK?) (Varchar)
Paint_Code (FK?) (int)
Paint_PGM (FK?) (int)
Pack_Value (Null)
Nametag_PN (FK?) (int)
Nametag_File (Null)
Nametag_Bin (Int)
Nametag_For (Varchar)
Nametag_Line1 (Varchar)
Nametag_Line2 (Varchar)
Nametag_Line3 (Varchar)
Nametag_Line4 (Varchar)
Nametag_Line5 (Varchar)
Maufnr (Int)
Partn (FK?) (Int)
Prodnet (Varchar)
Route_Group_Id (FK?) (Int)
Small_Tag_Label_Line1 (Varchar)
Small_Tag_Label_Line2 (Varchar)
Small_Tag_Label_Line3 (Varchar)
Status_Hold (Null)
Lgort (Varchar)
Ntgew (Int)
Gewei (Varchar)
Assembly_Sections (Null)
Dispatch_Active (TinyInt)
Last_Dispatch_Build_Type (FK?) (Int)
wo_genealogy
Id (PK) (Int)
Genealogy_Type_Id (FK?) (Int)
WO_ID (FK) (Int)
Serial_Number (FK?) (Varchar)
Pallet_Number (FK?) (Varchar)
Station_Number (FK) (Varchar)
String_Value (Varchar)
Numeric_Value (Int)
Tstamp (Datetime)
Position (Varchar)
Serial_No -> Serial_Number
wo_unit
ID (PK) (Int)
Wo_Id (FK)
Assembly_State (FK?) (Int)
Serial_No (Unique?) (Varchar)
Unit_Count (Int)
Wo_Id -> Id
Wo_Id -> Wo_id
wo_bom
Id (PK) (Int)
Wo_Id (FK? PK?) (Int)
Comp_Id (FK?) (Int)
Comp_Struct_Seq (Varchar)
Comp_Struct_Typ (Null)
Comp_Pt_Of_Use (Null)
QTY_Per (Int)
Identifier (Null)
Comp_Type(Null)
Comp_Ecn (Null)
Comp_Ecn_Cd (Null)
Status_Id (FK?) (Int)
Parent_Id (FK-ID?) (Int)