Cascadia Scientific: Equipment Health Improvements

September 2023 to February 2024

Contents

Background

What Cascadia Does

Cascadia Scientific, Inc. uses machine learning to help mining companies improve their practices. It starts with a small computer that the company manufactures, which get attached to mining trucks. These devices, known as PMMs, connect to the truck and to various sensors to collect data about the truck's fuel consumption. They then send this data to a server where it can be analyzed. Along with the actual devices, Cascadia also develops software for customers to view their data, as well as see various analytics and reports. Together, these products are known as the SmartRView system, and they allow mining companies to make better-informed decisions about their practices, to decrease fuel costs and their carbon footprint.

Components of the Cascadia SmartRView System

The entire physical SmartRView system is made up of multiple components:

- A PMM: this is the main computer of the system. Inside it has an antenna to send the data, and all other pieces connect to this device.
- 2 Flow Meters: these measure the fuel flow rates before and after the fuel flows through the engine – these give a very precise value for a truck's fuel consumption. These also have the capability to measure the fuel temperature through each of the meters.
- GPS/GNSS: this provides data about a truck's geolocation, including latitude and longitude, altitude, and attitude.

The system also connects to the vehicle's network to obtain data such as engine speed and payload weight.

Why Equipment Health is Important

Equipment is installed on the outside of the trucks and is exposed to the elements. Parts of the system can be damaged or disconnected. There are over 400 PMMs in use across mine sites worldwide, making it nearly impossible to manually inspect each piece of equipment regularly to confirm its health status.

There are ways we can see if something is not working by looking at the data that piece of equipment is sending. For example, if a flow meter has been damaged or is blocked, we expect the flow to be zero, but we can see that the engine is running on the truck. Thus, in most cases, it is possible to determine if a component of the system is broken just by inspecting the data it is sending.

Before the Equipment Health improvements began, the Field Service Technicians at Cascadia would have to manually check the data themselves to look for cases of equipment needing to be repaired or replaced. This activity is time consuming and prone to human error, and it becomes increasingly difficult as the Cascadia customer base grows. To combat this problem, we needed to develop an automated equipment health testing system that would be reliable and could provide technicians with a list of equipment that needs maintenance.

Before I began my work, many of these tests we already in place. But the results from them could not be trusted – they produced many false positives, sometimes because incorrect assumptions were previously made about how a malfunctioning piece of equipment would manifest itself in the data, and sometimes because of a bug in the code. The goal of my work specifically was to get these tests to a place where technicians could have confidence that there actually is a problem if one of the tests is failing.

Overview of the Equipment Health Tests

General Form of Equipment Health Tests

Each of the tests have a similar general form: analyze the most recent data, and if this data exhibits a certain behaviour, the test will fail. Upon failure, an Equipment Health Issue will be recorded in the company's database. If the given test passes for a piece of equipment, and if there already exists an open Issue for this test and equipment, the Issue will be closed.

Most of the tests analyze the last 24 hours of data that a truck has sent, but it can be up to the last week of data if the truck has not been sending data very frequently. If we have received very little data in the last week, the Equipment Health Tests will not run for that truck and its Equipment Health status will be Indeterminate.

Severity

Each test has one of four possible severities:

- Error: this indicates that a piece of the system will most likely need to be repaired or replaced.
- Warning: this indicates that there is likely an issue that requires attention, but it is not detrimental to the overall functionality of the system.
- Info: this indicates an issue that does not require attention but is something to keep an eye on. An Info can also mean a problem with the software that is causing the test to fail.
- Mitigation: this indicates that we have intentionally configured the system to behave a certain way, because an issue is causing bad data.

Assigning a severity to each test allows the employees at Cascadia to determine how urgent a problem is, and to prioritize more severe problems.

Making Changes to the Tests

Analyzing and Updating Existing Tests

The first step in this project was to go through the tests that had already been written to evaluate Equipment Health, to gauge how well they were performing. This step involved going through each test individually, and getting a list of devices that were failing each test. Then, I had to manually go through the data from that device to see if there did appear to be a problem.

The data was inspected using the debug feature in Visual Studio, and by stepping through the code to get a closer look at how the program was running. At times I also had to go to the database directly to view the raw data and confirm any trends or patterns there.

If a test was determined to be correctly diagnosing Equipment Health problems, no changes needed to be made. However, if it seemed to be incorrect, we needed to investigate *why* it was incorrect, and then determine how it needed to be altered.

When I started this manual inspection process, I did not have much knowledge about how the SmartRView system worked when attached to a truck, and how the data was expected to behave if the truck was healthy compared to if it was unhealthy. This step involved a lot of collaboration with Field Services Technicians to learn more about the physical system, including how devices were connected to each other, and how the PMM sent data to the server.

Adding Additional Tests

Throughout the process of improving the Equipment Health tests, we would occasionally find a case of broken or malfunctioning equipment that did not yet have a test to diagnose that issue. Tests were added as their need was identified, and the new tests followed the same general form as the original tests.

Optimization of Code

To run all the Equipment Health tests, we need to read in data from almost 20 different tables, for over 400 trucks, for one week. This equates to a few hundred thousand rows of data. When the tests were previously running, they did not require as much data, but with the addition of new tests, the large queries started to put a strain on the database.

The Equipment Health Batcher, which had been running hourly, had to be temporarily turned off while we addressed this issue. We made sure that all the tables in the database were indexed properly, which made the biggest difference in the overall runtime of the batcher. I also went through the code to double check that it was only ever actually retrieving data that would be used in the tests and were not unnecessarily loading data.

After this investigation, it was also decided that the program did not need to run every single hour, and instead it was sufficient for it to run every 4 hours. These improvements have stopped the Equipment Health Batcher from putting significant strain on the database.

Next Steps

Integrating Work Order Software

The goal of this project is to provide an automated system to detect Equipment Health issues and alert technicians when a piece of equipment needs attention. To further automate this process, we would like to combine this software with an already-established work order ticketing software. This way, when a problem is detected, a work order ticket can be automatically generated and assigned to a technician.

Providing a UI

Once the tests got to a place where they could be trusted, improvements to the UI for equipment health were made for customers to see some of the reports as well.

Appendix: Processes and Technologies Used

C# / .NET Core

The original tests were written in C#, using the .NET framework. To retrieve the data, Microsoft Entity Framework Core was used in many places in the code. Throughout this process, I learned how to write SQL queries and LINQ queries.

AWS Lambda, EventBridge, CloudWatch

The Cascadia software architecture relies on AWS to host many of its services, so it was essential for me to learn about many of these technologies. The Equipment Health program in particular runs on a Lambda in AWS, that is scheduled every four hours. I learned what a Lambda function is, how it works, and when it is appropriate to use a Lambda function. The Lambda "trigger" in this case is the time, so I also learned how to create a Rule using EventBridge as a way to trigger the function to run every four hours. Finally, to verify that the Equipment Health Bather Lambda was running at the expected times, I used the Log Files in CloudWatch to double check when it was running and for how long it ran.

Agile Software Development Process

As this is my first Coop term, I was able to learn what software development is like in the professional world, specifically in an Agile Software Development environment. The changes to the Equipment Health Batcher went through a series of tests done by Cascadia's QA Analyst, and bugs were found and fixed before the Equipment Health module was released to customers.