Better Tools for Fault Diagnosis in Complex Systems

Nicholas Reiter and Nathaniel Guy

University of Washington, Computer Science & Engineering/Aeronautics & Astronautics

PROBLEM

Determining the root causes of problems for complex electromechanical systems is difficult. Systems can detect fault states quite reliably using simulated software models; however, even when a fault is detected, it can be difficult to determine the underlying reasons, and resolution method, for that fault. Some anticipated faults may be automatically recovered from, but others are more complex and require humans to understand their root causes before resolving them.

MOTIVATION

The fault diagnosis process is very difficult, and can take weeks to months. It involves intense scrutiny of potentially thousands of data channels, and often the only comprehensive understanding of how these data channels relate to each other is encoded in human "tribal knowledge." For this reason, a large amount of domain expertise is required to even begin this process.

Because diagnosing the root cause of system faults can take thousands of man-hours of expert time, any tools that can facilitate the navigation and organization of this task can potentially save a lot of money and time. However, visualizations designed for telemetry monitoring and fault diagnosis encounter the following major issues:

- Displaying data from thousands of different channels isn't practical
- It's unclear how best to organize thousands of different interrelated channels to make interesting ones findable
- We'd like to have automatic discovery and visualization of relationships between channels
- The tools ought to be able show many views of data in a coherent interface

Modern research has made some headway on these issues, and we've attempted to incorporate some of their findings within our project. Cancro et al developed useful techniques for packing large numbers of channels into a dense rectangular space [1], and Yairi et al demonstrated ways to show change correlation between data channels [2], which were inspirations for our Global Correlation Matrix and Channel Correlation Vector. Simple fault detection methodology was adapted from Willsky [3].

Much of our organization and choice of components was influenced by personal experience with telemetry monitoring and analysis software used within the space industry, and the issues that we experienced first-hand when dealing with problem of fault diagnosis across huge datasets.

APPROACH

We incorporated many different ideas from the fields of data visualization, and examined historical tools and theories for ideas. In this section, we give descriptions and justifications of components we implemented.

Fault Monitoring Window

Borrowing from traditional monitoring interfaces, this part of the visualization has a traditional live data display panel, including the current system time and values of specified telemetry channels. Which channels' values are shown is configurable.

This window also carries a description of any recent faults, along with details of the rules that originally caused them to trigger, and any other additional fault-related notes that system designers or operators have included for reference. This additional information, uncommon in traditional fault monitoring system, accelerates the fault diagnosis problem by immediately pointing towards possible root causes. Finally, an LED-shaped icon shows the current fault state, in order to quickly grab the attention of the operator.

Channel Hierarchy Window

A degree-of-interest tree displays the hierarchy of all of the data channels. Major systems are broken into subsystems, which are then again broken into smaller subsystems, until the channels are reached at the leaf nodes. Clicking allows for expansion and navigation of the tree, and allows channel data to be selectively added to the Plotting Window. When a fault occurs, any related nodes on the tree are flagged, and those flags are propagated upwards to allow an operator to trace through the tree to find the channels affected by faults.

Plotting Window

This component provides a set of configurable plots of live telemetry channel data. The plots update as new data comes in over the network. This component is tightly linked to the Channel Hierarchy Window, which provides an interface for adding new channels to be displayed. When a fault occurs, the channels determined to be most relevant to that fault are shown. Even in a fail-safe mode where no new telemetry is being received, the plot display allows a human operator to review data leading up to the fault.

Global Correlation Matrix

This 2D matrix shows Pearson Correlation Coefficients across many different data channels, based on the most recent telemetry channel values. These cross-correlations are visualized using hue to show positive/negative correlation, and intensity to show the strength of that correlation. With this widget, an operator can see correlations of channel changes, suggesting possible interconnectedness or causation.

APPROACH (cont.)

Channel Correlation Vector

This component is similar to the Global Correlation Matrix, but shows channel correlations for a specific channel under review. The names of correlated channels are displayed for quick reference and faster lookup than the Global Correlation Matrix could provide.

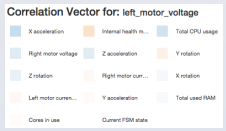
Additional Features

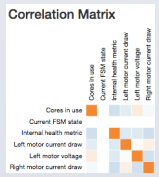
We implemented a number of additional changes to the application that differentiate it from traditional telemetry monitoring interfaces and improve its applicability to a wide variety of systems and problems:

- Channel hierarchy and fault definitions are easily configurable via .json files
- Data server performs automatic fault detection based on configurable rules
- Data server supports multiple simultaneous connections from clients
- Telemetry can be simulated based on channels means and standard deviations
- Support for multiple telemetry sources, including serial input, via highly modular components
- Reconfigurable telemetry deserialization format
- Plot simulation functionality

RESULTS

Images of our interface are depicted below, along with short descriptions.

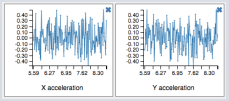




The Correlation Matrix and Correlation Vector use hue and intensity to show positive and negative correlation over time for pairs of channels.

The Channel Hierarchy Window depicts the interrelationship and organization of telemetry channels via an expandable degree-of-interest tree.

The Plotting Window displays a dynamically-generated time series of recently received channel data for channels of specific interest.



CONCLUSIONS AND FUTURE WORK

Our work showed that an extensible, yet generic, interface for facilitating fault diagnosis across large telemetry data sets is feasible, and simple knowledge about the data and its interrelationships can be conveyed very quickly through certain design choices. However, it remains to be proven if the tools provided are adequate for advanced data discovery with extremely complicated issues.

In the future, we plan to apply this system to actual aerospace systems and issues in order to iterate on its functionality and fix issues encountered by human operators using it for their work. Future extensions may include adding 3D visualizations for systems with clear physical analogs (e.g., mechanical systems), fault replay, and detailed plot annotation (both automated and human-initiated).

REFERENCES

[1] Cancro, G.; Turner, R.; Nguyen, L.; Li, A.; Sibol, D.; Gersh, J.; Piatko, C.; Montemayor, J.; McKerracher, P., "An Interactive Visualization System for Analyzing Spacecraft Telemetry," Aerospace Conference, 2007 IEEE , vol., no., pp. 1,9, 3-10 March 2007.

[2] Yairi, T.; Kawahara, Y.; Fujimaki, R.; Sato, Y.; Machida, K., "Telemetry-mining: a machine learning approach to anomaly detection and fault diagnosis for space systems," Second IEEE International Conference on Space Mission Challenges for Information Technology, 2006.

[3] A. Willsky, "A Survey of Design Methods for Failure Detection in Dynamic Systems," Automatica, 1976.

ACKNOWLEDGEMENTS

We would like to acknowledge the developers of the software tools we used to make our project:

- C3.js
- qTip
- Bootstrap
- jQuery
- Mike Bostock (D3 and various inspirational applets)

In addition, we would like to thank Dr. Jeff Heer and the teaching assistants of UW's Spring 2015 CSE 512 Data Visualization course.