REACH: REAL-TIME DATA AWARENESS IN MULTI-SPACECRAFT MISSIONS

Lori Maks, NASA/Goddard Space Flight Center

Jason Coleman, Aquilent, Inc.

Abstract

NASA's Advanced Architectures and Automation Branch at the Goddard Space Flight Center (Code 588) saw the potential to reduce the cost of constellation missions by creating new user interfaces to the ground system health-and-safety data. The goal is to enable a small Flight Operations Team (FOT) to remain aware and responsive to the increased amount of ground system information in a multi-spacecraft environment. Rather than abandon the tried and true, these interfaces were developed to run alongside existing ground system software to provide additional support to the FOT. These new user interfaces have been combined in a tool called REACH.

REACH—the Real-time Evaluation and Analysis of Consolidated Health—is a software product that uses advanced visualization techniques to make spacecraft anomalies easy to spot, no matter how many spacecraft are in the constellation. REACH reads numerous real-time streams of data from the ground system(s) and displays synthesized information to the FOT such that anomalies are easy to pick out and investigate.

Motivation

The size of the Flight Operations Team (FOT) of a typical NASA spacecraft mission is 3 to 5 people. This team is responsible for the daily operations of the mission, including monitoring and maintaining the health and safety of the spacecraft. In recent years, proposals have begun to appear for missions consisting of multiple spacecraft working together. These proposed "constellations" of 3 or 4—and sometimes as many as 100—related spacecraft would pose a dilemma. If the size of the FOT grew in proportion to the number of spacecraft, the FOT would quickly grow large and costly. On the other hand, if the FOT did not grow in proportion to the number of spacecraft, then the current means of monitoring health and safety information would prove insufficient for the sheer amount of data generated by the

spacecraft constellation.

GSFC's Advanced Architectures and Automation (AAA) Branch had previously developed the Visual Analysis Graphical Environment (VisAGE), a software package that coupled secure data distribution with advanced data visualizations. It contained many common visualizations like bar charts, strip charts, and scatter plots, but also incorporated advanced visualization techniques such as 3-D modeling of spacecraft instruments with color-coded state information. We also developed a custom visualization called the Data Carousel to provide summary information about hundreds of points while simultaneously allowing the user to "drill in" to more detailed info about any selected point. The AAA Branch decided to apply our visualization expertise to the domain of constellation health and safety.

The AAA Branch began a project to prototype a visualization system that could easily display the large amount of data related to a spacecraft constellation, and present it in such a way that a small FOT could manage an entire constellation. We had to determine how to present more data in the same screen real estate, yet in a way that improved its clarity.

Description of REACH

Our desire was to build a constellation visualization tool that would help the FOT quickly identify threats to the health and safety of their spacecraft constellation. To do this, the tool would have to receive telemetry data from a ground system, process that data in some manner, and then display the results in a way that would draw the user's attention to the most critical information. These requirements could be restated as three questions that we would have to address:

1. With the existence of so many different ground systems, and with the possibility that each spacecraft in a constellation may have its own ground system, how do we retrieve telemetry, limit, health, and safety data for a constellation?
2. Once we have the low-level telemetry data, how do we synthesize it to provide some higher-level meaning about how the spacecraft and subsystems are performing?
3. What visualization techniques can we use to present the resulting information to the FOT, and how do we draw the FOT's attention to the most critical information?

Any Given Ground System

We first had to decide how to develop our ground system interface. Ideally, we wanted REACH to be able to interface with *any* ground system so that *every* mission could choose to use it. However, a survey of

several ground systems in use at NASA revealed that none of them share a common data subscription interface. While several of them support connections over a TCP/IP socket, the protocol that each ground system uses to control the flow of data differs. As a result of these differences, custom code must be written for each ground system before REACH can connect to it.

There is movement at GSFC toward developing a standardized connection interface to be used by all future ground systems. Indeed, the development of standardized data format descriptions, protocols, and interfaces within scientific communities goes hand-in-hand with the push toward open source, plug-and-play solutions in software development. In years to come, the standardization of ground system interfaces will make it possible for REACH and tools like it to interact with *any* ground system without requiring custom code to be written. Based on current ground system architectures, however, that is not presently an option.

Given no uniform way to retrieve data from a ground system, we decided that our first release would support the ITOS ground system. ITOS (Integrated Test and Operations System) presented itself as the obvious choice since the AAA Branch had already developed an ITOS interface for VisAGE, and because several missions at GSFC that have been supportive of the REACH effort were using ITOS, including the Small Explorer (SMEX) team.

Synthesizing Information

The presence or absence of a limit violation tells the operator how a component is behaving with respect to its own physical parameters. That is, limits protect the physical extremes of the components by notifying the FOT when, for instance, the temperature of a component is too high or too low for the component to function properly and without damage. This approach provides component level information, but does not incorporate the operations concept of the spacecraft as a whole, including relationships between the components and with the environment. The demand on the batteries may be much greater, for instance, when the spacecraft is holding its station and performing science than when the spacecraft is maneuvering and its instruments are not in use. The minimum amount of voltage required is different in each of these situations. An unsophisticated limits system can only choose one of these values as the "low voltage" limit. We wanted to reinterpret the limit information from the ground system so that limits were adjusted based on the context of what the rest of the spacecraft was doing.

When a limit violation first flags, the FOT scrambles to figure out why this violation has occurred, what the impact to the mission might be, and what corrective action they can take to bring the value back within limits. If some telemetry value begins to violate its limits frequently, then over time the FOT

comes to regard these recurring violations as normal. FOT personnel use limit violations to get a "feel" for how a spacecraft is doing on each pass: some mnemonics are flagged as limit violations, but the experienced team member knows that this particular mnemonic has been acting flaky lately, and it's not a big deal. This comfort with a spacecraft in the presence of limit violations takes time and experience to acquire, as well as knowledge of how the telemetry and limits interact with each other. We provide a simple mechanism for senior FOT members to codify this knowledge in the REACH Mission Model Engine.

We wanted a user to be able to look at a single indicator that would communicate the status of the entire spacecraft in one glance, without having to look at all the limit violations for that spacecraft and then determine by "feeling" whether that spacecraft was safe and healthy. This single indicator would "roll up" all of the limit violation information for the spacecraft into an overall, yes-or-no appraisal of whether the spacecraft was "healthy" or not.

A "healthy" component, whether that component is a spacecraft, a subsystem, or a telemetry source, would be one that is functioning properly within its own physical parameters, and is also functioning as expected based on its relationships to other spacecraft components and the current state of the spacecraft. The health of a component is represented as a percentage from 0 to 100.

The astute observer might note that we've traded a limit system that used just 5 values (red low, yellow low, green, yellow high, red high) with a system that has 101 values, from 0 to 100 inclusive. This allows for much finer specification of health. For display purposes, we still keep red, yellow, and green as color-coded severity indicators, but green might be used when the health of a component is above 80%, yellow when it's above 50%, and red below that.

REACH uses a Health Model Engine to calculate the health of every component and subsystem in the spacecraft. Using the information in the mission database, the Health Model Engine constructs a model of the constellation. It can be thought of as a tree with a single root node: the constellation itself is a node containing one node for each spacecraft; each spacecraft node contains a node for each of its subsystems; each subsystem node contains one node each for all of its associated telemetry. By evaluating the presence or absence of limit violations in the raw telemetry, we can assign a "health" to each subsystem node. We can then assign a health to each spacecraft node based on its subsystems. Finally, we can assign an overall health to the constellation node based on the health of all of the spacecraft.

Most of the work on the Health Model Engine dealt with how to calculate health values from the information that was already present in the ground system and the mission database. In this way, REACH can begin working immediately out-of-the-box, without any unique configuration (i.e. if any parameter is

within red limits the health of the entire spacecraft is flagged as red). Ideally, we would like the FOT to build a hierarchy showing which telemetry mnemonics belong to which subsystems and how various subsystems connect to each other. To support this, we've developed an editor environment where FOT personnel can create equations and relationships linking the health of a node to other nodes. Users can drag-and-drop components into equations, make dependency relationships, or add custom scripts to interact with an expert system or other third-party tool. The end result is that the users can build a description of their constellation in hours or days, not the weeks or months required by an expert system. And, as they grow accustomed to seeing a certain mnemonic flag as a violation over and over, they can easily adjust their Health Model equations to recognize this violation as harmless and exclude it from determinations of the spacecraft's overall health.

Visualization Techniques

The desire to create powerful visualizations to help the FOT easily evaluate the health and safety of a spacecraft constellation was the driving force behind REACH development. The original concept was to develop a visualization that would allow the FOT to decide, in one glance, their priorities regarding every limit violation or unexpected condition in the constellation. In practice, developing one visualization to communicate all of this information was impractical. An important feature of an effective visualization is simplicity, and one visualization that can communicate everything about a constellation of 100 spacecraft will not be simple. We managed this complexity by creating several visualizations: each simple, and each tailored to communicate a different type of information about the constellation.

Before explaining the visualizations we developed, we should discuss the visual techniques upon which they are based. Our software bridged two schools of thought about data visualization. The first school says that *all* relevant data should be available to the user at all times. The second says that only the most important data should be shown to the user and the rest eliminated to reduce the clutter and increase the clarity of the visualization. We employed each technique in different visualizations. The combination of the two schools has been effective.

First, we developed a number of visualizations that were concerned only with high-level status. We stated earlier that our Health Model Engine "rolled up" information. These visualizations deal with that rolled-up information. For example, the Data Abacus provides information about the stability of a data point over time by moving multiple "beads" back and forth on a bar, like an abacus. As the bead, representing the current value of the data point, moves back and forth, it leaves a thick trail behind it. Thus, it's easy to tell how much the overall health of the spacecraft has been fluctuating by observing the length of the

bead's trail. Communicating *why* the value is fluctuating is not the job of this visualization; it's job is to make the FOT aware of the fluctuation.

When the Data Abacus is used to display the overall health of each spacecraft in the constellation, this fluctuation, or *trending,* becomes very useful information because it gives the user an idea of how the spacecraft as a whole is behaving and how its subsystems are working together.

Another visualization that hides less important data for the sake of clarity is the Mission Grid. The idea behind the mission grid is to associate data at the spacecraft and subsystem level. This information is laid out in a grid where each row represents a spacecraft and each column represents a subsystem. The data used to represent each row and column is "rolled up" from the telemetry lying below that spacecraft or subsystem. By looking across a row, the user can see the health of the subsystems of a particular spacecraft; by looking down a column, the user can see how that particular subsystem is performing on each spacecraft in the constellation. This view—comparing the performance of a single subsystem across every spacecraft in the constellation—is a unique benefit of REACH.

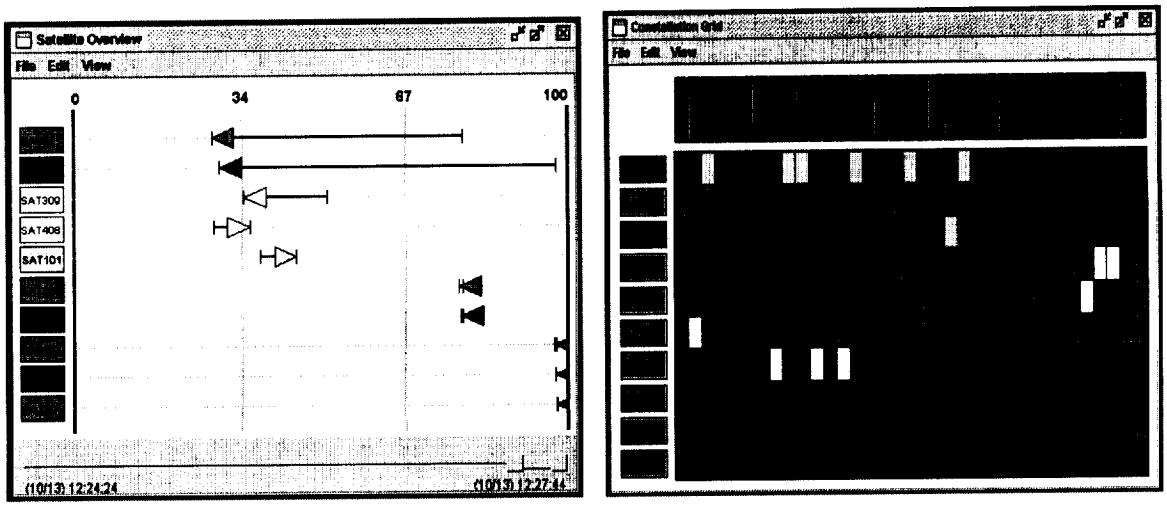

Figure 1: Data Abacus and Mission Grid

These "high level" visualizations were descendants of the VisAGE visualizations that the AAA Branch had developed. On the other end of the spectrum, the visualizations the AAA Branch developed in conjunction with the Human-Computer Interaction Laboratory (HCIL) at the University of Maryland, College Park, are designed to give the overall context of the mission without hiding *any* health and safety data. This is done using a technology developed at the HCIL called a Zoomable User Interface (ZUI).

ZUI technology is used to speed a user's traversal of information in situations where there is a large quantity of data. In its *resting state,* a ZUI will provide a survey of all data in the system at a very low

resolution. The user can use this to get the context of the entire system. In the realm of constellation flight operations, this resting state might be a view of the high-level health of every spacecraft in the constellation. When the user finds a reason to investigate—maybe the health of one of the spacecraft has dipped to a critical level—then the user can *zoom into* that spacecraft and see more detailed information about it. A good example is the ScatterPlot ZUI shown in Figure 2. It is displaying the health of every telemetry point in a constellation of 100 spacecraft! The list of spacecraft names on the left is displayed so small that it's difficult to read, but this does not keep us from seeing the dots representing the health of every telemetry point associated with every spacecraft. The telemetry points are plotted on a table ranging from 0% health to 100% health. Even though we can't read the names of all of the spacecraft, it's easy to see which ones have telemetry points with red limits (below 33% health). Seeing these values in red limits would cause the user to move the mouse over them and thus "zoom" the information about this spacecraft to a sufficient size to be read. The look-and-feel of this technique is similar to dragging a magnifying glass across the screen.

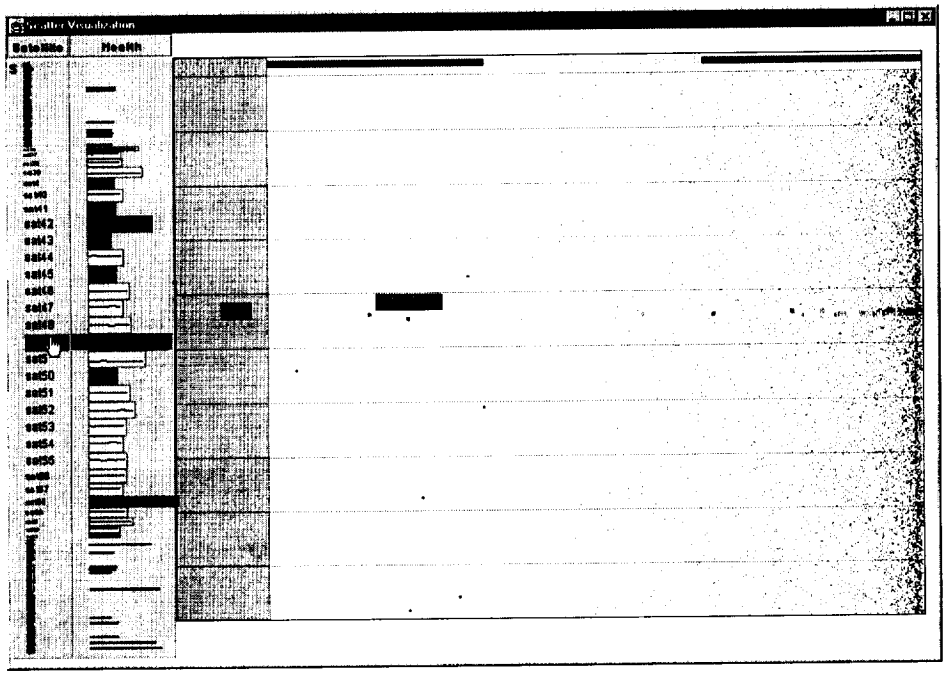

Figure 2: This Scatter Plot is an example of a Zoomable User Interface

The sampling of REACH visualizations shown here use size, location, and color[1] to indicate the importance of some aspects of the data, while minimizing others. The benefits of these visualizations become obvious when using them alongside a traditional ground system. High-level visualizations help the FOT see what's going on in the big picture, while ZUI-type visualizations help the user keep from losing track of low-level information.

Conclusions

REACH was designed to support constellation missions of up to 100 spacecraft. Feedback from early prototypes has indicated that it can also be useful in other multi-spacecraft environments such as:

- The Small Explorer (SMEX) program and other similar programs which consist of multiple spacecraft operated from a single control center, and
- A consolidated mission operations view of health and safety status data from otherwise independent missions

Our goal in undertaking the development of REACH was to provide a tool to make the health and safety data from a potentially large constellation of spacecraft manageable by a small FOT. We have developed a means of calculating health that is intuitive and user-configurable. In the future, when mission databases grow to include more metadata about relationships between spacecraft, subsystems, and telemetry, the amount of FOT intervention in the creation of the Health Model will greatly decrease. For now, the efficiency and level of detail of the basic Health Model that runs 'out-of-the-box' are improved a great deal when the FOT spend a few hours refining the model by hand. Lessons learned from this and other Expert System activities indicate that a tool must be simple to configure and not require a significant up-front investment, for it to be accepted by the FOT.

We have developed visualizations that are visually compelling and that provide an impressive synthesized view of current spacecraft health and safety status. In the future we want to focus less on developing new visualizations and more on enabling interaction of the FOT with the REACH visualizations and Health Model engine. REACH visualizations make users aware of health and safety problems, but REACH was

[1] Using color to carry information is often frowned on in the world of data visualization because it is not accessible to users with color blindness. Historically, however, ground system displays have made use of the colors red, yellow, and green to display limit information. For the sake of familiarity we decided to follow that custom. Along with color we use other, more accessible, visual techniques to avoid any situation in which a difference in color is the only form of notification to the user that something important is happening.

not intended to provide a means of issuing commands to the ground system. Observations so far have indicated that when the FOT see a problem they want to have flexibility in their means of diagnosing it, yet REACH's current suite of visualizations lock the user into a specific form of inquiry. We have usability tests scheduled with the SMEX FOT to investigate alternative solutions.

REACH has had three important releases so far, culminating in the current version: REACH 1.0. You can learn more about REACH, download in installer, and discover other exciting products of the AAA Branch at http://invision.gsfc.nasa.gov/avatar/.

Acknowledgements

This work was supported by the NASA Goddard Space Flight Center. The author would like to thank the following individuals:

- Julie Breed, Branch Head, Advanced Architectures and Automations Branch (Code 588)
- Maureen Madden and Patrick Crouse, GSFC SMEX Lab
- Ben Bederson, Human Computer Interaction Lab (http://www.cs.umc.edu.hcil/)