Telemetry Metrics: Monitoring Data Quality in the Spacecraft Ground Data System

Robin A. O'Brien

April 4, 2006

During the launch of Mars Odyssey, ground data system (GDS) engineers experienced a glitch in the ground data system that caused us to re-evaluate how we looked at spacecraft telemetry, particularly during the spacecraft development period and for critical spacecraft events in flight. Spacecraft telemetry told the subsystem and instrument engineers about the health and status of the spacecraft, but there was surprisingly little information about how well the ground data system was doing in getting information from the spacecraft to the engineers.

The problem for the Mars Odyssey launch was with a single channel not updating as often as expected. Spacecraft engineers considered calling off the Launch, but eventually decided that this particular channel did not provide information that was crucial for launch. It was only after the post launch acquisition of the Odyssey signal that ground data system engineers heard there had been a concern about the channel.

While the spacecraft engineers were celebrating a successful launch, the GDS engineers were busy investigating the loss of a channel. We pinned down the culprit fairly quickly – the table that Flight Operations was using to extract channels from the raw telemetry was for an earlier version of flight software! So when the ground system tried to extract a channel, it would occasionally run into an undefined channel, at which point it marked the record to indicate that it had given up trying to extract channels from that particular packet and moved on. There was an error message to the operator's terminal, but it was lost in a vast sea of operational messages.

The Case for the Ground Data System

The ground data system generally ranks dead-last in the hierarchy of any spacecraft project – what Project Manager wouldn't want to take a little funding from their GDS and put it toward their spacecraft? Yet the loss of the Mars Climate Orbiter demonstrated that even the humble ground data system could contribute a great deal to the loss of a mission. The incident at the Odyssey launch showed the potential impact the ground data system could have during mission critical events. And during the Assembly, Test and Launch Operations (ATLO) phase, every mission uses the ground data system to diagnose serious problems in their hardware and flight software.

While it seems obvious that missing telemetry data is a bad thing, it has become even more of a bad thing as spacecraft design has changed over the last decade. With missions that used Time Division Multiplexing, channels were sampled and transmitted at regular intervals – if you missed one, another one would come along soon. With the rise of packet telemetry, information is much more event-driven. Information is still repeated, but the possibility of missing a single event is now higher and much more dependent on the quality of flight software. And of course, there is more data. The Mars Reconnaissance Orbiter (MRO) has a maximum rate of 6 megabits. The number of channels within spacecraft telemetry is also going up – XX for the Mars Exploration Rovers, YY for MRO. Again, the question we need to ask is ***How well is the ground data system doing in getting information from the spacecraft to the engineers?*** This type of monitoring must begin in the pre-Launch phase

as part of a concentrated effort to verify that both the spacecraft and ground data system function as needed.

In response to the table mismatch at the Mars Odyssey launch, during the ATLO phase of the Mars Exploration Rover the GDS developed a relatively simple program, the ***Data Quality Monitor***, to monitor for potential communication problems between the Flight and ground systems. The way ***DQM*** reported telemetry metrics was also influenced by the ATLO experience; the possibility of a GDS engineer getting a call in the middle of the night from a test engineer complaining that they weren't seeing any data. We needed information that a test engineer could easily find and relay over the phone, so we could pinpoint the problem and go back to bed as soon as possible.

While DQM was successful for MER, the Mars Reconnaissance Orbiter came along and showed us shortcomings of the ***DQM*** and our telemetry metrics. When you are working with a spacecraft that is capable of sending down more data than all previous Mars missions combined, you get some surprises. With its 6 megabit data rate and simultaneous Ka-band and X-band capability, MRO has challenged the ground data system more than any other mission at the Jet Propulsion Laboratory in the last 20 years. The following list of telemetry metrics is a direct result of the "Lessons Learned" from these Mars Missions.

Telemetry Metrics

Data Heartbeat tells us how whether data is coming into the system. This seems obvious, but there is one potential area of confusion. The requirement should be to have a relatively steady report interval, even when the data rate is changing dynamically and drastically; at 550k I'm happy to get a heartbeat message every 8000 bytes, but if the data rate suddenly drops to 120 bits per second, I certainly don't want to wait until I get 8000 bytes to get a heartbeat message. On the other hand, if the system is barreling along at 6 megabits, I don't want a heartbeat every 8000 bytes – no need to see the workstation dump out 93 messages a second. This requires being able to set some either/or type requirements on heartbeat reporting – tell me when I get N bytes **or** however much data you got in the last two seconds, whichever occurs first.

Sync State History. Once the data Heartbeat messages assure us that the ground data system is indeed receiving data, the GDS engineer needs to know the current status of the telemetry? Are we receiving valid telemetry (Insync) or junk (Outasync). If the system is Insync, are we getting idle data? That would explain why the spacecraft engineers are complaining that they aren't seeing data, but indicate that the ground system was performing as expected. For the Mars Reconnaissance Orbiter we have found it useful to know which virtual channel was currently being downlinked; if a science virtual channel were coming down it would explain why no one was currently seeing any engineering data.

Some systems combine the Sync State History and the Data Heartbeat information, letting you know with each insync or outasync record your current status. GDS engineers then found themselves frequently going back through perhaps thousands of heartbeats to know when the state of the system changed. It was the information on things changing that was really needed. If you sat down at a workstation and looked at ground data system software that reported that the system was currently "Insync", you'd almost always ask yourself "Well how long have I been insync? Two hours or two minutes?". Particularly if you'd just been called because users were experiencing data outages. For the MER and MRO Projects we set the up the ***Data Quality Monitor*** tool so that the "States" were configurable. The records of interest – those that indicated a "state" had changed – appeared for the first time, the State, current time and other information were reported then on. DQM did not report anything else until a different record that appeared in our "States" file was received. This history of changes was often invaluable when analyzing problem reports in the system. The Sync State History

was a relatively small report that could quickly tell you when the system experienced data outages.

Packet Sequence Anomalies need to be monitored, but they are not always the packet gap that the GDS engineer worries about. CCSDS tells us that for a given APID the packet sequence counter will increment by one or rollover from 16383 to 0. Any other behavior is a packet sequence anomaly. The spacecraft can have all kinds of legitimate reasons for creating a packet sequence anomaly, and it generally takes some knowledge of the spacecraft activities to help explain all the anomalies. But our goal is to know how well the ground is communicating with the spacecraft, so some anomalies are more "suspicious" then others. A gap that spans a very short period of time and covers just a few packets is unusual; mode changes and operational problems typically cover a bigger span of time and/or packets. During flight of course we expect to see packet gaps due to signal loss, but people are often surprised at how often they occur prior to launch. In testbeds, packet loss without any corresponding change in the sync state (dropping sync) usually indicates a problem in Flight Software or the Ground System – and of course the Ground System would need to be the first area examined for failures.

During pre-launch testing for the Mars Exploration Rovers (MER), I&T engineers started reporting that they were not seeing the changes they expected in certain channels. Investigation showed that individual users were not receiving all the data coming from the spacecraft, and that the amount of data missing could vary widely between workstations on the same local area network. For years, JPL engineers had relied on receiving broadcast data, which was based on UDP. While the protocol did not guarantee that data would be delivered (as in TCP/IP), performance over the years showed negligible loss, and setting up to receive broadcast data was simpler for users than a point-to-point connection. This all changed with MER; 8 analysts could be sitting at workstations monitoring the spacecraft during pre-launch tests, and only half of them might see certain conditions reported by the ground system. The exact cause of this is still something of a mystery. Some people thought the new network switches were to blame, since loss on some of the older networks seemed smaller. The size of the telemetry packets being transmitted has also been linked to the problem. For MRO, the spacecraft team actually decreased their packet size for some engineering records; they now feel that data loss for broadcast data is at acceptable levels.

Not having this metric available leaves the Flight Project vulnerable. The Deep Impact Project used the telemetry packet counters as a Flight Software routing field; consequently, there was no way easy to determine that packets were being lost in the ground system. If MER had not discovered the broadcast problem, Deep Impact may have been left with some true data mysteries to solve.

Decommutation Errors Decommutation is the process that extracts actual telemetry measurements from the data. These measurements (channels) are typically the unit of data the spacecraft engineers are most interested in – what's the temperature of the left solar panel? Is the latch opened or closed? There our two kinds of communication problems that can be caused by decommutation errors. The first is the loss of telemetry. When the JPL Ground Data System detects an error in the decommutation process it stops decommutating that record and marks it with an error flag. So any other channel measurements in that particular record are never extracted – this was the case during the Mars Odyssey Launch. The second problem is incorrect channel values. Channel values are not completely defined in the actual telemetry; JPL missions use tables in the Ground Data System to determine which channels occur where in the telemetry. These tables are typically generated by Flight Software or take inputs from Flight Software, so new tables are typically set up for each flight software delivery. The wrong version, or a buggy version, means that channels may go out which are incorrect. The decommutation error isn't noted until it hits a condition the software can't handle – a channel in the data isn't found in the table, or the packet length is shorter than the tables indicate, so the decommutation can't be done. But channels prior to the error

detection may be available via the ground system and received by the spacecraft engineer. So a misconfiguration of the Ground System can have the same impact as a bug in Flight software. This was the problem that inspired us to create the ***Data Quality Monitor***.

MRO – Latency and Volume

The Mars Reconnaissance Orbiter, launched in August 2005, has a maximum downlink telemetry rate of 6 megabits, which is unprecedented at the Jet Propulsion Laboratory. The SIRTF mission has a maximum downlink rate of 2 megabits, and the other Mars missions had maximum rates that were well under 1 megabit. During MRO's Assembly and Test Phase we were constantly being challenged with the MRO GDS choice – lose data or fall behind?

Prior to launch, the serial interface between the spacecraft and the ground data system is the critical juncture between the two systems. The serial interface operates at the rhythm of the spacecrafts data rate; the ground system must be able to read the data from the serial interface at that rate. If the ground system is not there to read the data as it is being written by the serial interface, the serial interface will drop the data into the proverbial bit-bucket never to be seen again. This is the dreaded "Data OverRun" condition.

At a 6 megabit telemetry rate, there was not horsepower to spare on the workstations and networks that made up the ground data system. While the baseline system appeared to handle the system in some instances, we found that it was operating at the edge of our performance requirements. A burst of outasync data or a jump in the percentage of channelized engineering could break the chain of applications used to process the data, causing a Data OverRun. To handle such bursts, we increased the size and complexity of the buffers we used in the ground data system. The problem with this was that these bursts would cause the ground system to "back up" from the spacecraft engineers perspective, and the data they watched would be time-tagged further and further behind the current time. While spacecrafts in flight typically present data at least as old as their one way lighttime, preLaunch this is not considered acceptable. MRO had one instrument that was so sensitive to thermal conditions in one test that they determined they could not fall more than two minutes behind in the telemetry, or they wouldn't be able to send commands to avert some catastrophic condition. And spacecraft engineers were understandably leery of not knowing the current condition of a spacecraft in test. So falling behind was not an option during ATLO and more workstations were added to the ground data system to handle backlogging. Data latency became the next telemetry metric that the Ground Data System needed to monitor.

Surprisingly, data latency was still an issue even after Launch. The telecommunication lines between JPL and the stations of the Deep Space Network could not accommodate all of the telemetry generated by MRO in realtime. So JPL developed a plan to send only the engineering component, isolated in a number of virtual channels, across the lines to JPL in realtime. The remainder would flow in as the data lines allowed (within some maximum limit of course). The majority of that 6 megabit data was science data; the Project said that the engineering component would be a steady 35 kbits at most data rates, well within the range that JPL typically processed for Flight Projects.

Eighteen days after the MRO Launch, the MRO ground system engineers found out that their latency problems were not over with ATLO. The spacecraft was downlinking telemetry at 550 kbits – a bit high but within the range of normal operations. And spacecraft engineers watched as the data being delivered to them fell further and further behind the current time. It was originally reported that we fell 20 minutes behind. On examination however, we found that when we went back and measured data latency (the difference between when the bits first arrived on earth and when telemetry channels were actually extracted and presented to

the users), we found that we had climbed to a maximum of 53 minutes behind in processing telemetry in realtime.

[Graph Here]

What could have caused this delay? MRO had downlinked at 550kbits before without this happening? The answer, and our next telemetry metric, is volume.

While the MRO downlink rate was not unusual, what was unusual about this particular incident was the volume of engineering data coming down. The flight ground data system had been set up with the understanding that realtime engineering would be around 35k, even though MRO would be downlinking at a higher rate, and this had been true prior to this. This particular incident was caused by a backlog of engineering data onboard the spacecraft, which was downlinked during this 53 minute period. We looked at the downlink rate and saw the same 550k rate, but failed to notice that the actual volume of engineering data had shot up, taking up the entire bandwidth. As the graph above shows, as soon as the volume of engineering data fell, latency began to drop.

Data Volume is an important consideration for all Projects however, particularly in ATLO, because it can measure how "flight-like" a given test is. For a mission like MRO, operating at the edge of their performance margin, bit-rate does not tell the entire story. Just because we got through one test at 6 megabits, does not guarantee success at the next test because of the percentage of idle data in the data stream. For example, Test A could run at 6 megabits with 60% of it's downlink composed of Idle frames – this is often true in ATLO where sequences are under development and the "TBD" areas are filled with Idle data. Test B, running at 6 megabits with only 10% Idle frames, has the potential to stress the ground data system much more. The bit-rate is useful information for the GDS engineers, but the volume of actual engineering data that makes up that bandwidth is the real number we need to be concerned with. Ideally, any data quality monitoring applications are going to measure volume over different time frames (and hopefully those time frames would be set-able parameters). You want to look at volume over relatively small periods as well as the span of entire tests, so you can see volume "spikes" in the telemetry flow. In the graph shown for the MRO incident, volume was measured over a minute.

Information Layering

How information is presented to users is at least as important as ***What*** information is presented to users. The decommutation errors that occurred during the Mars Odyssey Launch were detected by software. Operators just did not spot the relatively unobtrusive message behind the huge barrage of information that was coming in to them. If operations personnel have trouble with these displays, what chance does a spacecraft I&T engineer have when they exercise ground software and flight software in the Project Testbeds? Is a ground data system engineer required for every test? Not if the ground system provides a ***layered*** set of information.

Ground data systems are typically designed for ground data system engineers; the applications give a huge amount of information that the end user is not really interested in. We don't really care what the engine temperature is when we drive our car – we just want to know if the temperature is too high. Our car dashboards don't scroll messages by that tell us that all car doors have terminated their closure procedures successfully; it just rings a little bell if we have left one open. Our car is not a bad model for the interface the ground data system should be giving to the majority of its users. It's a pretty complicated piece of

equipment that has had its information reduced to the minimum that the user needs to know about. Most spacecraft engineers I have watched operating in Project Testbeds will bring up the ground data system as required and then immediately put it in the background, never to be looked at again. They'll look at displays for the spacecraft information they are interested in, but they are typically not interested in information about the ground data system (unless they have problems). I've often wondered if we could reduce the system to a single Idiot Light:

Green: Running Just Fine
Yellow: You may be missing some data
Red: Go get a Ground Data System Engineer.

Of course, a ground data system engineer would want to know much more about the system, particularly if called in the middle of the night. The application programmers would want to know even more about what their individual application was doing. All three levels of information are needed, but too much information may be as obfuscating as too little information (as it may have been during the Mars Odyssey Launch). This information should not be combined in the same one-size-fits-all display; it should be layered for the different classes of users.

At the low end of the information scale, we need a bare minimum of information about the ground data system for the non-ground data system users (spacecraft and flight software engineers who run the ground data system as part of their operational testing). This could be as simple as the Green/Yellow/Red Idiot light code mentioned above, but there is a good argument for including some kind of heartbeat information here as well. Our cars give us feedback by moving (or not), and users need to know that the ground system was ok the last time it was checked at hh:mm:ss to give that green light any meaning. The only other real telemetry metric these users need is that one or more of these metrics has a problem, as indicated by the "Yellow" or "Red" condition of the ground data system. This is analogous to the alarms actually set on spacecraft data and might benefit from a similar implementation.

At the high end of the information scale, we have low level application information that the application developers put in for monitoring the current state of their application. Ironically, this is probably the most common type of information we see in the ground data system and the least used. Information in this class would be application specific and give little insight into the status of the system as a whole. It is certainly helpful to the software developer creating the application, but its use to the higher level user is limited. This area is most likely to lose the critical error message among a sea of status messages.

Telemetry metrics fall in the middle of the scale. They are related to the status of the ground data system rather than a particular application. They are of low-interest to the non-ground data system user (but of greater interest if the system does not appear to be working correctly). Within this area, there should be another set of layers: summary and history. During critical mission events, GDS engineers monitor the ground system just as the spacecraft team monitors the spacecraft subsystem. They need to see the current status of the telemetry areas discussed; the current number of gaps in the data, the number of decommutation errors observed, the data latency right now. But for troubleshooting system problems after they've occurred (and they are often not obvious until after they've occurred) the ground data system engineers need to review the history of telemetry metrics: show me all the gaps that occurred between 12 and 1pm. When did we go out of sync on March 28th? Were there any decommutation errors during the test? Pity the poor mechanic who only gets hearsay that the car was making funny noises last week; the GDS engineer, given a well defined ground system that records telemetry metrics during processing can have a much clearer view of what happened to the system, even when it's been unattended.

The ***Data Quality Monitor*** was an attempt to isolate and monitor key information about the quality of communication between the spacecraft and the ground system in a very simple interface. We learned from MRO that latency and data volume are measurements that should be added to this list of telemetry metrics. The experience on JPL's various Mars Missions has been that ground data system engineers need information ***about*** spacecraft telemetry just as the spacecraft engineers need telemetry about the spacecraft for the success of the mission.

DRAFT