

# A Whale of a Tale: Creating Spacecraft Telemetry Data Analysis Products for the Deep Impact Mission

Kathryn F. Sturdevant<sup>1</sup>, Jesse J. Wright<sup>2</sup>, Yee N. Lee<sup>3</sup>, and Roger A. Lighty<sup>4</sup>  
*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109*

This paper describes some of the challenges and lessons learned from the Deep Impact (DI) Mission Ground Data System's (GDS) telemetry data processing and product generation tool, nicknamed "Whale." One of the challenges of any mission is to analyze testbed and operational telemetry data. Methods to retrieve this data to date have required spacecraft subsystem members to become experts in the use of a myriad of query and plot tools. As budgets shrink, and the GDS teams grow smaller, more of the burden to understand these tools falls on the users. The user base also varies from novice to expert, and requiring them to become GDS tool experts in addition to spacecraft domain experts is an undue burden. The "Whale" approach is to process all of the data for a given spacecraft test, and provide each subsystem with plots and data products "automagically." Since spacecraft team members receive their data in a comprehensible format, they can immediately begin their analysis, or load the standard formatted data into familiar tools. The Whale generated data products became so critical to understanding the state of the spacecraft during pre-launch tests, that a parallel version of Whale was created to run daily in operations. During the Deep Impact cruise phase from January to July 2005, in addition to the daily "ops" whale, the four testbeds were often running 24 hours a day/7 days of the week and most of the telemetry data produced during those tests were processed through Whale. To compound the issue, there were three possible sources for data: two spacecrafts (flyby and impactor), and the flyby data was dual, meaning it could come from spacecraft side A or side B. Data analysis products therefore had to be created separately for each of the three sources of data: SCU\_A, SCU\_B and IMP. The lessons learned from Whale will be used to design the next generation Testbed and Data Archival systems for future JPL missions.

## I. Introduction

THE "Whale" tool was initially envisioned as an archive utility for testbed data, however it soon mushroomed into a product generation utility, with archive as a secondary feature. The nickname "Whale" was indicative of the size of the task. Being a joint mission between Ball Aerospace in Boulder, CO and the Jet Propulsion Laboratory (JPL) in Pasadena, CA, half of the subsystem engineers were not familiar with the existing JPL proprietary ground system. Additionally, some users were accustomed to having a team to perform the product generation service for them, whereas the existing system expected users to generate their own products using a given set of utilities. Thus emerged the need for an automated means of providing telemetry data products to subsystem engineers. The challenges faced in building such a utility included network security issues between the two organizations, each with their own security requirements, and processing a large amount of data in a time-efficient manner. Once the basic network and data processing were in place, the requirements on Whale continued to grow, as Whale became the primary means of analyzing project data. Whale needed to adapt as the project's needs changed through normal mission phase progression. Test data sets became significantly larger, and more capabilities were required. A parallel operational version of Whale was created to support spacecraft operations, in addition to the original Whale,

---

<sup>1</sup> Ground Data System Engineer, Integrated Ground Data Systems Section, 4800 Oak Grove Drive/M.S. 264-214.

<sup>2</sup> Ground Data System Engineer, Flight Software and Data Systems, 4800 Oak Grove Drive/M.S. 264-214.

<sup>3</sup> Network/System Administrator, Enterprise Computing and Networking, 4800 Oak Grove Drive/M.S. 311-104.

<sup>4</sup> Export Technical Representative, Office of Export Compliance, 4800 Oak Grove Drive/M.S. 180-202.

which continued to support the testbeds. The Deep Impact project thus relied on the Whale tool through the pre-launch, launch, cruise, and impact phases of the mission.

The Whale tool suite demonstrated the power of scripting both existing and new utilities to form an extremely useful tool for spacecraft missions. The end result was a reduction in the amount of time and effort required of subsystem engineers to get their data products, thus allowing them to analyze their data sooner and with less effort on their part. Once the network connectivity issues were resolved between the various testbeds and the Whale machine, creeping requirements, including processing larger data sets, became the main concern. Due to inconsistencies in collecting the test data, part of the Whale task became a manual review of the data products to ensure successful completion. Although the Whale utility was envisioned to be both a data processing and archival system for spacecraft data, due to the resource limitations and short lifespan of the project, most of the effort was applied to data processing while the archival function was limited to procuring more disk space. This paper will outline the design of the Whale utility, the problems encountered, the chosen resolutions, and make recommendations for improvement.

## **II. Whale Technical Challenges and Resolutions**

The main requirement of Whale was to have a single repository for processing and archiving spacecraft testbed data that was shared between Ball Aerospace and JPL. This function was provided by a dedicated machine with a large disk. Test conductors would submit their data to the Whale task, which would process the data to create plots and other consumable products, then store the data and results in a central repository. Users would be notified via email when data products were available for analysis, and a web link was provided in the email for convenient access to the data.

Whale itself was comprised of Perl scripts, which incorporated a Perl-based telemetry format library, a plotting tool, a GNU ghostscript utility, a web browser, and email. The Perl-based telemetry library was developed by a previous task at JPL for the purpose of making a proprietary data format useful to other programs. The GNU ghostscript utility was written to support the Whale task as it concatenated single plots in PDF format into one file.

This section will address the primary issues that concerned Whale which include: Network Configuration, Whale Data Flow, Parallel Processing Design, Personnel, Requirements, Whale Data Products, Processing Time Issues, CPU and Disk Usage, Data Growth Through Mission Progression, Email List Management, along with Whale Results Archival and Presentation.

### **A. Network Configuration**

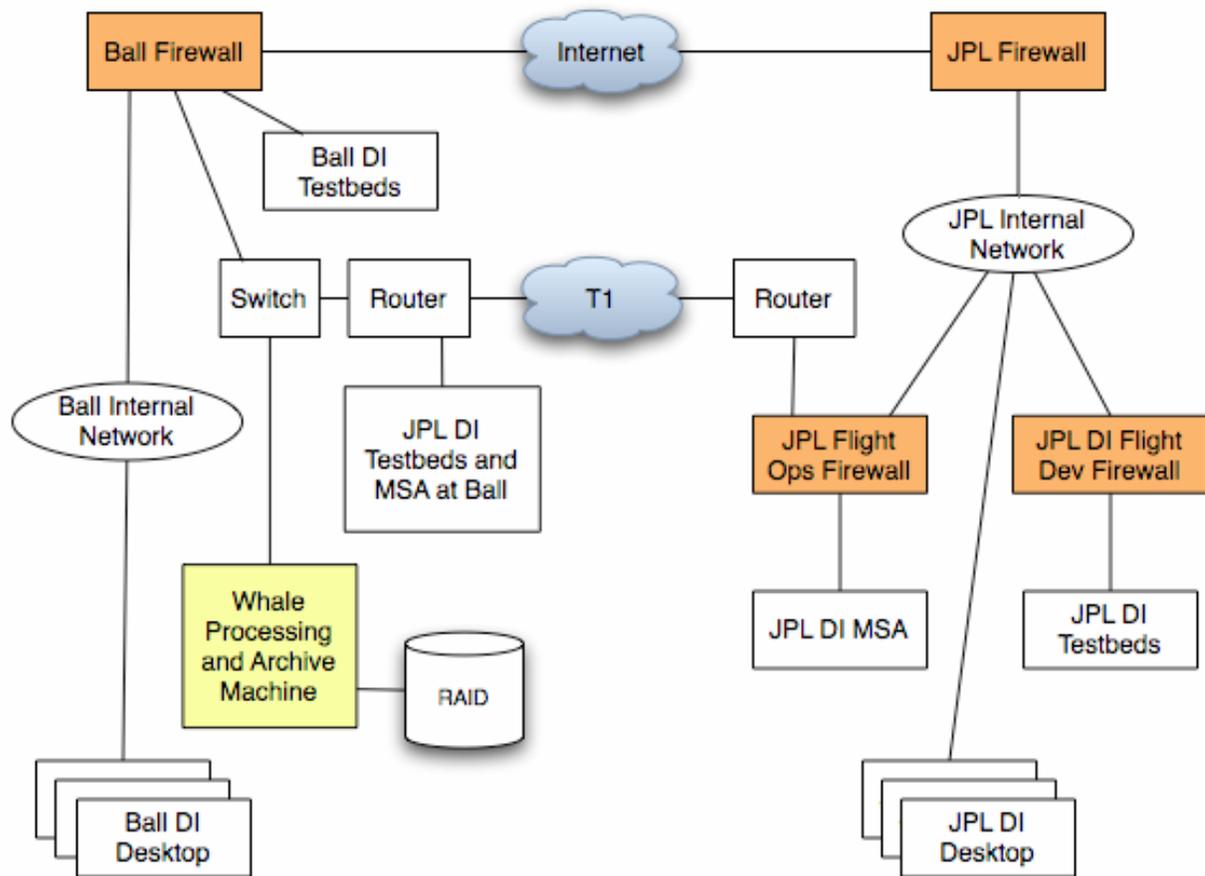
The initial challenge was to architect the network for data flow and accessibility between the Ball Aerospace and JPL organizations, each with its own networks, firewalls, and security requirements. In addition to the firewall to get into JPL, there were separate internal firewalls protecting the testbed and operational machines. Testbeds existed both at JPL (on the JPL Flight Ops LAN) and at Ball Aerospace (on the JPL Flight Ops LAN via dual T1), and all of them needed the ability to submit data to Whale. Additionally, users at both organizations needed access to view the resulting products. Therefore, testbeds, mission support areas (MSAs), and some desktop machines all needed access to the Whale machine.

The initial plan for the Whale processing machine was for it to have two Ethernet cards to allow it to be both on the JPL and Ball Aerospace networks. Security for both organizations did not support the idea, so it was decided to make Whale machine its own "island," with the JPL Flight Ops (DI) Network and Ball Internal Network having access. The Ball firewall was configured to allow access from specific internal Ball (DI) desktop machines. The JPL Flight Ops router was configured to allow access from JPL Flight Ops Networks such as Ball/DI testbeds, JPL/DI Mission Support Area (MSA), JPL/DI Flight Dev. Testbeds, and others. When the testbeds were installed at JPL, they were put behind their own protective firewall, which initially prevented the JPL testbed machines from connecting to the Whale machine.

In order to automate the submission of tests, testbed data needed to be transferred to the Whale processing machine via scripts. The JPL firewalls initially prevented automated data transfer between JPL DI Testbeds and the Whale machine, and human intervention was needed to perform the copy between the firewalls. JPL Security had to approve a firewall exception waiver to allow JPL testbed machines direct access to the Whale machine before automation could be implemented. Once the firewall access issues were resolved, all testbeds had access to the Whale machine, and the architecture was thus made satisfactory from a submission perspective. The data access perspective, however, still had inequities for JPL versus Ball users. The Whale machine was physically located at Ball, and their network was configured to allow certain desktop computers direct access to the Whale machine. The data link provided in the Whale email was helpful to them as they had direct links to the data products, which were

archived on the Whale machine. JPL users did not fare as well, as personal desktop computers are not allowed on the Flight Operational Network. Therefore, they would receive the Whale email notification, however the data link did not work for them. Their access the data via the web data was limited to machines located inside the firewall, so they had to run a browser remotely on a machine behind the firewall, but displayed on their desktop machines, which was very slow. Alternately, they could transfer the data files, but that was a two-hop process. Users therefore required accounts on the processing machine so they could transfer the data directly to their desktop machines for analysis. This defeated the initial plan to allow only limited login access to the repository machine.

The following diagram shows a simplified view of the Deep Impact network at Ball and at JPL.



**Figure 1. The DI Whale Network Layout**

The network architecture design must provide security, data transfer, and data availability. This becomes especially difficult when agencies are teaming on a project and have their own security requirements and need equal access to the data products. The majority of users wanted the data products available on their desktop machines for viewing convenience or so they could perform more processing on the products, such as reading comma-separated-value (CSV) products into Excel, or running their own utilities.

Another network-related security issue critical to the Whale design was how the telemetry data would be transferred from the testbed machines to the Whale machine for processing. The testbeds were on their own networks, each of them writing to a database, and the Whale machine was on a separate network. Ideally, the Whale utility could have queried the databases for the data; however, Network security did not allow the Whale machine to query over the network. The tests were therefore configured to save all of their data to files in addition to writing to the test databases. The files were then submitted to Whale for processing. As tests grew from 8 hours to 72 hours, so did the data volumes, so the file transfers became significant.

This design presented a network performance issue. The JPL Testbed access to the Whale server was via T1 with a bandwidth of 1.54Mbps, whereas the Ball Testbed access to the Whale server was via LAN with a bandwidth of 10/100Mbps. In order to minimize the network bandwidth bottleneck, the Whale server placement should have

been decided based on the estimate of the amount of test data that each of the testbeds would generate over the life of the mission. In the case of Deep Impact, the load shifted from Ball to JPL testbeds after launch and that was not considered when placement was decided. Moving the Whale server from Ball to JPL post-launch was considered, however, with daily Whale runs, and often multiple daily runs, there was never enough down time to make that possible. The JPL team thus had a much less convenient and more cumbersome access to the archived data, due to network slowness and network configuration, which did not allow them direct access from their personal desktop machines.

## **B. Whale Data Flow**

The Testbed Whale Data Flow process was defined as such:

1. From a testbed machine, the test conductor would initiate the Whale submission script and be prompted to enter the test id number and directory location. (Other “metadata” required for test processing were saved in a file that was created automatically in the test environment and included information such as the testbed number and flight software version.)
2. The Whale submission script would execute as a setuid script as the Whale login. It would compress and tar the contents of the directory into a file located in the Whale login account. It would then secure copy the tar file across the network to a staging directory on the Whale processing machine.
3. A cron job was configured to start the Whale processing script periodically. (The script could be started manually when necessary.) It would check the input queue for waiting tests.
4. For each test in the queue, the Whale processing script would create test subdirectories in the web directory based on the testbed number, test name, and test date. It would then untar and uncompress the data and begin processing.
5. The event data would be run through a script that converted it to ASCII text.
6. The sensor data would be turned into an “expanded” comma separated value format (ECSV).
7. Plot products would be generated by the plotting tool “Galleryplot” using the ECSV and a list of sensor identifiers.
8. CSV products would be created from the ECSV and used to generate table products.
9. The plots would be concatenated into subsystem-specific files.
10. An email notice would be sent to the Whale email distribution list with a link to the archived test data and its Whale products.

The Operational Whale data Flow was similar to the testbed Whale, however, the data first had to be collected:

1. The operational data was queried from the flight database. A Whale query tool was initiated via cron at the end of a day of year (DOY). It would query all three data sources from the operational database (SCU\_A, SCU\_B, and IMP) for sensor values and events, and create a corresponding metadata file to imitate the testbed data format. For consistency, the operational data was assigned a “testbed number” for processing, however the web interface identified it by name to avoid any user confusion.
2. The query script would then use the Whale submission script to transfer the data to the operational Whale processing machine for staging.
3. The cron job to process the operational Whale data was executed once daily.
4. The Whale processing steps for operational data were then the same as the Testbed steps for items 2 through 10.

Please refer to the following diagram for the Whale data flow.

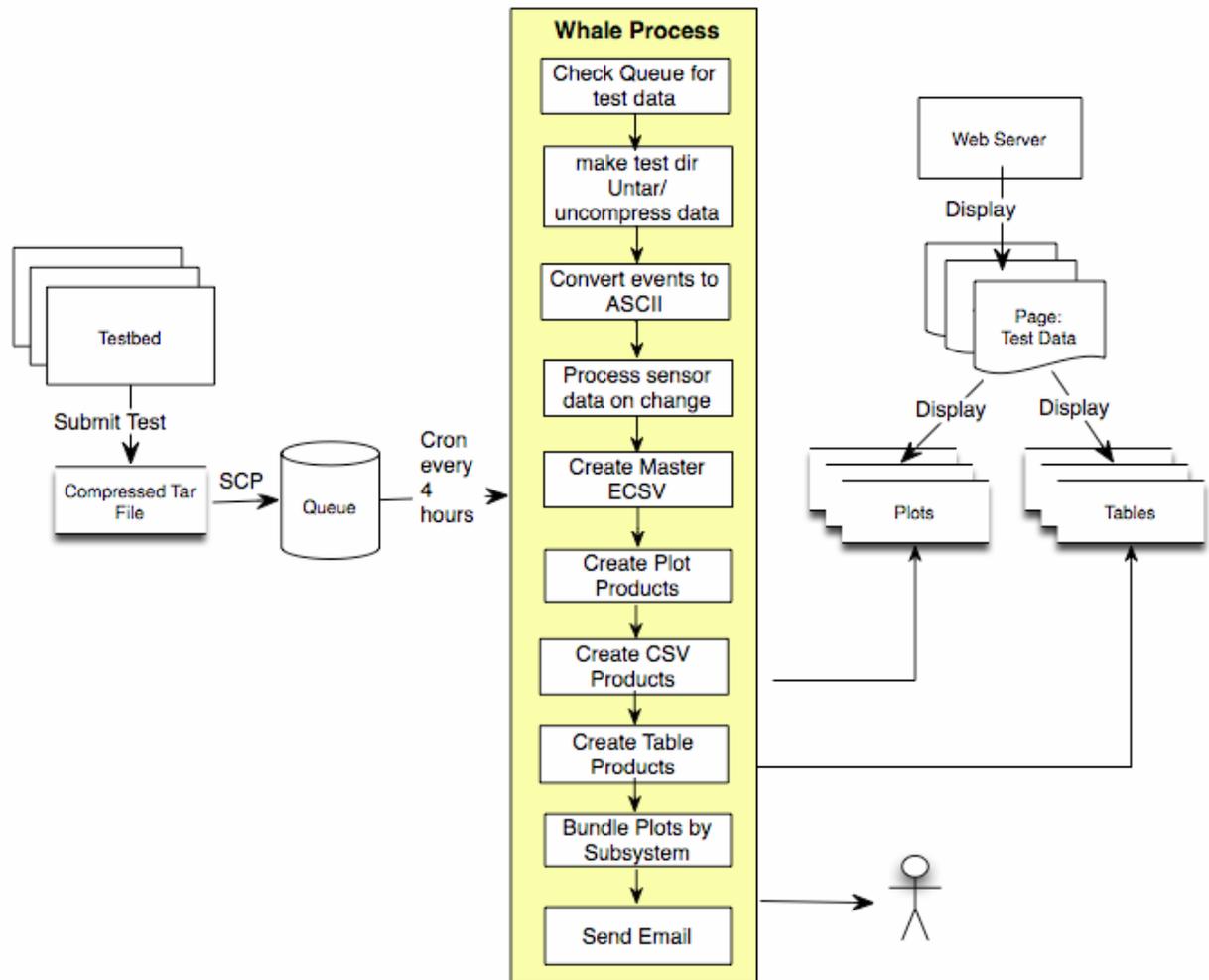


Figure 2. The DI Whale Data Flow

### C. Parallel Processing Design

Another key design issue to the Whale task was how to process the data in a time-efficient manner. In the first pass through the telemetry data, we opted to compress the data by sampling only on change. On change means taking only the instances when the data points changed values, as opposed to on sample, which would mean every time the value was reported. On change sampling reduced the data to 1/10th of the previous size. The data were then expanded to comma separated value form for processing. The data consisted of millions of samples of more than 20000 different sensor values. A plot and a table needed to be created for each sensor value's samples in isolation. In an early prototype version of Whale, running our plot and table creation tools on the base data file serially created run times of 2-3 days for 8 hrs worth of data. Clearly, this was not satisfactory. The solution was to write a script that would read the base file and create up to 250 files of selected data. That tool was run over the base data file as many times as was needed to extract all of the information. The limit of 250 was due to our UNIX configuration, using Perl scripts, which could have only 256 open file handles. This cut the run time from days to hours. The resultant files were sent to plot and table formatting software. When the plot product generation changed to use Galleryplot, our homegrown Gnuplot wrapper, this preparation step was no longer necessary.

As launch approached, the demand for faster turnaround of data products grew. The number of sensor values was significant, eventually reaching 21,710 between the two spacecrafts, flyby and impactor (IMP). The subsystems each provided a list of the sensor identifiers in which they were most interested, and Whale was modified to run on the reduced list of 3709 sensor ids. The reduced list was sufficient throughout cruise and impact, with subsystems providing occasional updates to their lists. Those subsystem lists were merged to form the processing sensor id list,

which was used to create the Master ECSV file. The subsystem lists were used individually to bundle the plots into subsystem specific files.

#### **D. Skeleton Whale Crew With A Wide Range of Users**

The initial version of Whale was created by one developer in his spare time. Four months prior to launch, with the tool not meeting project expectations, more resources were allocated for its development. Whale was allocated a combined total of two full-time employees four months prior to launch. Whale directly supported more than 40 users between JPL and Ball Aerospace. While one of the main drivers for creating Whale was to support the novice users, experienced users benefited as well and Whale became the primary source of data products.

The Whale team was initially funded only through launch, but the demand for data products and the dynamic nature of the testbed environments warranted the continuation of Whale support throughout the life of the mission. Creeping requirements and unexpected inconsistencies with testbed data caused the Whale utilities to have to be constantly updated to meet changing conditions, and tests routinely would need human intervention in order to be formatted properly prior to processing. The dependency on Whale products led to its adaptation to process operational data. Operational support was intended to last 30 days post launch, but it continued to the end of the primary mission, which was 6 months. It is anticipated that Whale will continue to be used by the Deep Impact project should it be granted funding for an extended mission.

#### **E. Requirements: Known Versus Creeping**

Four months prior to launch, a basic Whale had been implemented, and was run manually on occasion per user request. The volume of complaints made it clear that the user expectations had not been captured as requirements, and that there was a break down in communication between the subsystem engineers and the Whale developer. The user plot requirements had to be gathered, followed by the types of data to process and the structure of the test data directories. With subsystem engineers busily preparing for launch, Whale requirements were difficult to gather and information crept in over time. This caused the Whale development to be exceptionally dynamic as often a new data source was mentioned shortly before a test was run that would require immediate Whale processing. For example, initial tests contained only flyby side A (SCU\_A) data, but soon flyby side B (SCU\_B) data was added, later followed by the impactor (IMP) data. Eventually, one, two, or all three of these data sources could be present in a given test data set, and each source required separate processing and products. The Whale utility needed to be able to recognize the existence of the three data sources in a test and process each one individually.

When the test conductors ran the tests, the data was saved to files, and the entire test directory was submitted to Whale. There were occasions when the directory structure was not known to the Whale developer until the data was submitted. At that point, the Whale tool had to be modified quickly to handle the new data format and process the data. In the months leading up to launch, it was not unusual for the new data format “requirements” to be gleaned from reviewing the test as it was submitted to the Whale processing queue. The Whale utility would be updated accordingly and the new Whale was then tested using that newly submitted test data. This was not “dangerous” in that the original test data still existed, but it is mentioned to show the dynamic, fast-paced environment that the Whale developers encountered.

Creeping requirements were a way of life and Whale had to adapt to meet the demands of the project as it approached each milestone, the months prior to launch being the most critical. Due to the last minute nature of the development, some problems were given quick fix solutions that did not take future use into account. Some of the decisions that were made early on caused problems throughout the project lifecycle, and would require manual intervention to resolve. For example, when “dual” tests were introduced to Whale (dual tests being those containing both SCU-A and SCU-B data), the test conductors (who had pressing schedules) did not want to modify their working procedures, so they decided they would manually separate the SCU-A and SCU-B data prior to submitting a dual test to Whale. This caused problems down the line as test conductors would either forget, or new test conductors would be unaware of the requirement, and the mixed data would not be discovered until the data products were reviewed and found to be incorrect. The Whale Team would then have to separate the data manually and reprocess the test. In retrospect, having the Whale utility separate the “dual” testbed data would have saved the mission a fair amount of reprocessing time during the life of the mission.

Inconsistencies were also introduced by differences in test procedures between different test conductors, often along the lines of JPL versus Ball. Additionally, tests were being performed 24/7 and test conductor stress and fatigue would introduce inconsistencies, or they would forget to perform an action that would require manual intervention by the Whale team to prepare the data for processing.

These creeping requirements and inconsistencies prohibited the testbed Whale tool from being a hands-off automated tool; rather it often required human intervention. A Whale team member was assigned the task of

reviewing each completed test for correctness in an effort to catch problems, hopefully before the subsystems were required to review the results. The operational Whale was much more automated as the data was queried and formatted by a separate utility. It needed manual intervention during times such as flight software updates, as the data had to be processed separately according to the version of flight software.

Some requirement changes were not necessarily caused by omission, but rather tool limitations. Plots were initially created with lines-only and the cause of a slope was indeterminate—was it interpolated or did it reflect a change in the data? The Gnuplot utility was using interpolation, whereas users plotting data in Excel would see stair-stepped plots. For some subsystems, the difference between a real slope and an interpolated slope was significant. Whale plots were thus modified to plot both data points and lines to alleviate the confusion. This demonstrates one of the trade-offs for increasing performance by processing sensor values on change versus on sample. The lines produced by on sample data points could be very different than those with on change and the subsystem engineers need to be made aware of the methodology applied in processing.

#### **F. Whale Data Products**

The Whale data products included Gnuplot PDF files, one for each sensor id requested in value versus time format. An additional step was to concatenate plots into files using a GNU Ghostscript utility based on lists of sensor ids provided by the subsystems. Each subsystem was provided one or more bundles of plots as PDF files.

The other data format provided by Whale was a table format, suitable for loading into other utilities, and the comma separated value format (CSV), which was often loaded into Excel. There was also a utility to convert data products that contained spacecraft events into an ASCII file.

The original plan was also to allow subsystems to define plots to contain more than one sensor id, and Gnuplot had that capability, however the multi-sensor plot enhancement was continually superseded by higher priority tasks.

#### **G. Further Processing Time Issues**

Due to the late development of the Whale task, a subsystem engineer produced a similar tool that gained a small, but vocal following and the project levied this processing as an additional requirement on the Whale team. This alternative approach required a duplicate pass through the data utilizing the “on sample” methodology as opposed to the faster “on change” methodology for product generation. Because the alternate methodology used a shorter list of sensor ids, the net effect was that it roughly doubled the processing time for each test. As tests grew in size, further complications ensued as the base utility for the “on sample” processing had a file size limit, which was reached, as DI tests grew longer. The Whale team had to separate the data manually and reprocess it in pieces in order to create the products for this subtask.

CPU competition also became an issue. While tests were being processed, users could be submitting new tests, some of which were quite large, or they could be copying other test results from the Whale machine to their desktop machines. As longer tests were being processed, some spanning for days, the delays due to competing tasks, and data transfers in and out, became more of a concern. Reprocessing incorrect tests put even more demands on the CPU.

An 11-hour test with 13 MB of data from one source took under 5 hours to process. A 28-hour test with 381 MB of data from two sources (SCU\_A and IMP) took 26 hours to process. Each data source was processed separately to create its own set of data products. If multiple tests were being processed simultaneously, the processing time was increased. This processing time included the “on change” processing with the Whale product creations (CSV, tables and plots), the plot concatenation script, and the second pass through the data using the “on sample” processing methodology.

#### **H. CPU and Archival Disk Usage**

The Whale machine was a Sun Ultra 2000 with dual CPUs, initially configured with 2-880 GB RAIDs. The project phase from launch to impact was 6 months long and data was growing at such a rate that running out of disk was feared likely. The project then purchased a 2.8 TB RAID to ward off any disk space issues prior to impact. The operational Whale machine was a Sun Ultra 60 with 2- 539 GB RAIDs. Total disk usage post impact was 3051 GB for the two Whale machines combined, testbed and operational. This total includes only data stored on the Whale machines, and does not include users’ Unix accounts or the data they copied to their desktop computers. Duplication was an issue as users resisted archiving any old tests and most wanted to download their own copies of test results to their desktop machines. Again, the network architecture and security requirements essentially required JPL users to get their own copies as they did not have equitable access to data products.

The short life of mission combined with a prior project data mishap caused the project to require that all data be available on line and archives were not trusted. Buying more RAID was a workable solution for a short-lived

mission, but might not be practical for longer missions. The initial plan was to develop an archival system where older tests would be written to DVD before being deleted from the file system.

### **I. The Ripple Effect of More Data**

The processing demands of Whale grew significantly post launch, as test runtimes increased from 8 hours to 72 hours. Additionally, post launch, the operational data was being processed daily on the same machine. A separate machine had to be acquired and configured to process the “Daily Ops Whale,” so testbed and operational data processing would not impede each other. Processing time for testbed data continued to rise as testbeds were often running to full capacity, 24 hours a day/7 days a week, at up to 200 kilobits per second. The project began to require a prioritization mechanism on test processing, which was a new requirement on Whale and had to be performed manually, such as by shutting off the cron processing to prohibit any new tests from running until the priority task was complete, and getting the users’ cooperation to refrain from submitting lower priority tests until critical tests had completed. The Whale team was often asked to initiate the test submissions and/or process them in a specific order.

Most tests ran between 8 and 20 hours regularly, and some operational readiness tests would run for a few days. Due to the large file sizes that would result with the multi-day tests, the test conductors had to break off their tests at a logical point, and submit sections to Whale, and then continue with their tests. This would allow Whale to provide products in a timelier manner, as subsystems could begin their analysis of one part of the test, without having to wait for the entire test and its subsequent Whale processing to complete. As tests were run, the data results were collected in the user’s home accounts. User disk space also had to be monitored and cleaned up significantly prior to the beginning of a long test, or we risked running out of disk space in the middle of a long test.

As data files grew, problems rippled through the system including processing time, disk space, and even the web browser groaned. Whale components would break under the strain and have to be fixed quickly. When files reached 2 GB, Whale had various problems. Whale results were accessed via browser, and Netscape was unable to display files that were larger than 2.1 GB so test directories would appear empty to users. The Whale team had to watch for oversized data files and compress or move them to correct the web display. Additionally, Whale was initially implemented to use tar and zip to transfer data from testbed machines to the Whale machine. Zip failed for files > 2 GB so Whale had to be modified to use tar with the compressed flag. As the project prepared for encounter, Whale broke again when the staging area disk allocation was exceeded-- 3 large testbed runs had been submitted into the Whale queue at the same time. The staging area was allocated more disk space and Whale continued.

### **J. Data Validation Based On Timestamp**

The main impediments to successful automated processing of testbed data sets were data files that were incorrectly collected by the test conductors. Data would be flowing before the spacecraft clock was initialized, there were time discontinuities between test transitions, and there were procedural inconsistencies between the various test conductors. Non-initialized clock times could be easily stripped by Whale, however, if the test conductor did not properly start and stop the data collection, test data from a previous test could be written at the beginning of a new test. Because the clock times were valid, the data was not obviously bad, and the problem would not be noticed until the plots came out “broken.” At such time, the Whale team reduced the data files manually to the time range specified by the test conductor and reprocessed the test.

Whale was unable in general to be smarter about tests with time discontinuities because it had a requirement to use Spacecraft Event Time (SCET), which is not unique in the testbed environment, as opposed to Earth Receive Time (ERT), which is. Tests were routinely repeated, so many tests could use the same SCET time range. Modifying Whale to search the entire data set for time regressions prior to processing could have detected this problem, but would have cost more processing time for all tests, most of which did not contain regressions. This would have been a significant cost as tests increased in length.

Please note that tests needed “metadata” to be appropriately processed. Test name, testbed number, flight software version, data source (SCU-A, SCU\_B, IMP) were saved in a test file for GDS debugging purposes, and the Whale tool made use of that information for processing.

### **K. Email List Management**

The Whale task made use of the institutionally provided email list server. This proved extremely useful to the task as it allowed users to subscribe and unsubscribe via email. The challenge was then to get users to understand the importance of subscribing and not view it as “spam.” Posting colorful “apartment for rent” style notices around the project area with tear-off tags for the email address proved very useful in getting users signed up. The list worked very well once everyone was signed up for it.

## **L. Results on Web Page**

The web page was designed to display the tests based on the testbed number, drilling down to the test name, then further drilling down to the test date. The Whale email provided a link down to the date level. The date directory contained the test data along with the generated Whale products. Later in the project, people were requesting to see tests processed by date. An index was being created manually initially, but fell by the wayside. A better approach would have been to create and update the date list as part of the processing, so users would have the option to access tests either way.

When the operational (OPS) Whale processing and archive was moved to a dedicated machine on the JPL network, the Ball users lost their desktop machine access to the products. They complained loudly and the response was to mirror the data between the OPS Whale Server at JPL and the Testbed Whale Server at Ball.

Due to the configuration of the networks and the bandwidths described previously, the access to the Whale repository from JPL was significantly slower than the access from Ball. Although there were users in both locations, the primary operational location was JPL and it also had the largest user base, so the configuration caused the majority of users to be inconvenienced. Proper upfront requirements could have eliminated, or at least reduced, the access issues experienced by the JPL users.

## **III. Conclusion**

The Whale utility grew from being a data archive utility to being the primary test data product generation tool for the Deep Impact mission. It played a critical role in supporting the subsystem engineers from pre-launch to impact, as it became the primary method for the DI team to view their test data and products. The project as a whole would have benefited greatly had the Whale requirements been properly defined early on, and then redefined as the project's needs became better understood. The redefinition of its primary function along with the continual demands to update its functionality caused the Whale task continually to be reactive rather than proactive. Although the basic internal Whale processing was well understood by the Whale developer, the majority of the user requirements gathering began when the project was already in a critical phase. The enormity of the Whale task was not fully recognized early on, and the ripple effect of its incomplete requirements was felt throughout the life of the mission.

Limitations in the network design made data gathering, transfer, and accessibility more challenging for the Whale task and the Whale users. Had the user access requirements and testbed processing demands been better reviewed at the beginning, the testbed Whale machine would likely have been located at JPL, thus improving access times for a majority of the users. With current security restrictions, it is unlikely that many desktop machines would have been granted access to the Flight Ops LAN, but had desktop delivery been a requirement, Whale could have delivered products directly via email. Additionally, the importance of having realistic data storage requirements for the project along with a reliable archival system cannot be stressed enough. The Whale team and the System Administrators were tasked with continually monitoring the disk space, purchasing and configuring more disks, and moving large volumes of data to balance partitions. The Whale utility had growing pains of its own, as it was not designed to process the multi-day tests and had to be adapted on the fly to keep up with the increasing size of the data sets.

Processing time trade-offs were made, and Whale implemented a parallel-processing approach along with the "on change" methodology in order to process large volumes of data and generate results a timely manner. Because this method affected the product results, it was important that the chosen methods be understood by GDS and subsystem engineers alike, to ensure that data product results be analyzed accordingly. The Whale products were well-received once the new plot requirements were adopted, though a subset of the users were already dependent on the competing tool's products and the Whale utility had to spend extra processing time to create those products. The competing tool would not have been built had the Whale task been properly defined and staffed in order to meet the projects needs in a timely manner.

In the end, Whale was a mostly automated system that needed routine review and maintenance to make up for the lack of requirements and the inconsistencies in the testbed data collection. The "automatic" generation of products was an improvement over previous systems, and could have been even better with more time and funding. The lessons learned in the development and maintenance of the Deep Impact Whale utility will be very useful in the development and design of future ground system utilities.

## **Acknowledgments**

K. F. Sturdevant would like to thank Hal Uffelman of the Jet Propulsion Laboratory for his plotting utility "Galleryplot" which is a convenience wrapper to Gnuplot, Jonathan Weinberg of Ball Aerospace for his "binall" script that merged multiple plot PDFs into one file using GNU GhostScript, and Lori Nakamura of JPL for her

contributions to Whale. Additionally, K. F. Sturdevant would like to thank the System Administrators, Yee N. Lee and Thomas A. Mckenzie of JPL for their tremendous assistance with network and system support.

The work described in this paper was conducted at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.