Lessons Learned from Daily Uplink Operations during the Deep Impact Mission

Joseph Stehly

California Institute of Technology
Jet Propulsion Laboratory, Pasadena, CA 91109, USA

The daily preparation of uplink products (commands and files) for Deep Impact was as problematic as the final encounter images were spectacular. The operations team was faced with many challenges during the six-month mission to comet Tempel-1. One of the biggest difficulties was that the Deep Impact Flyby and Impactor vehicles necessitated a high volume of uplink products while also utilizing a new uplink file transfer capability. The Jet Propulsion Laboratory (JPL) Multi-Mission Ground Systems and Services (MGSS) Mission Planning and Sequence Team (MPST) had the responsibility of preparing the uplink products for use on the two spacecraft. These responsibilities included processing nearly 15,000 flight products, modeling the states of the spacecraft during all activities for subsystem review, and ensuring that the proper commands and files were uplinked to the spacecraft. To guarantee this transpired and the health and safety of the two spacecraft were not jeopardized several new ground scripts and procedures were developed while the Deep Impact Flyby and Impactor spacecraft were en route to their encounter with Tempel-1. These scripts underwent several adaptations throughout the entire mission up until three days before the separation of the Flyby and Impactor vehicles. The problems presented by Deep Impact's daily operations and the development of scripts and procedures to ease those challenges resulted in several valuable lessons learned. These lessons are now being integrated into the design of current and future MGSS missions at JPL.

Nomenclature

autonav	=	auto-navigation
ASP	=	Automated Sequence Processor
CCSDS	=	Consultative Committee for Space Data Systems
CFDP	=	CCSDS File Delivery Protocol
DKF	=	Deep Space Network (DSN) Keyword File
DSN	=	Deep Space Network
FSW	=	Flight Software
JPL	=	Jet Propulsion Laboratory
MGSS	=	Multi-Mission Ground Systems and Services
MPST	=	Mission Planning and Sequence Team
PEF	=	Predicted Events File
Rad List	=	CFDP Radiation List
SASF	=	Spacecraft Activity Sequence File
SCMF	=	Spacecraft Message File
TRF	=	Transaction Request File

I. Introduction

THE Deep Impact mission to comet Tempel-1 climaxed with some of the more spectacular science results ever collected by a spacecraft. However, the process to command the spacecraft proved to be a challenge to the entire operations team. This paper presents an overview of the process used prepare products for uplink and the lessons that were learned from this process. First, the different types of uplink products are discussed followed by how the spacecraft was commanded. Next, the uplink product development cycle is described along with the

tools developed to facilitate that process. Finally, several lessons learned are presented as well as how they have been implemented into current and future MGSS missions. The purpose of this paper is not to describe the development decisions made by the Deep Impact Project, just how the MPST tools made the uplink process workable and the lessons that were learned from operating the spacecraft. Due to export compliance regulations several sections of this paper have omitted technical details that portray the intensity of the mission. Every effort has been made to preserve the overall challenge while still complying with regulations.

II. Uplink Products Overview

Deep Impact used two distinct uplink product types to command the spacecraft. The first of these was the traditional spacecraft message files (SCMFs). The SCMFs consisted of all real-time commands that were radiated to Deep Impact's Flyby vehicle. The remaining files that were sent to Deep Impact were the Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP) products. These commands and files are discussed below.

A. Real-time Commands

Real-time commands were radiated to Deep Impact as a binary SCMF. A spacecraft immediately executes real-time commands once they are received. Deep Impact had a variety of real-time command classifications that were used. Real-time commands were used for such tasks as activating sequences, loading tables, and requesting telemetry from the spacecraft.

B. CFDP

CFDP is a standard file delivery protocol that provided Deep Impact with a means of transferring files from the ground to the spacecraft. CFDP transfers could be reliable, meaning all components of the uplink are verified as successfully reaching the spacecraft, or unreliable where the uplink occurs without final verification. Examples of CFDP products used by Deep Impact were sequences, parameter tables, FSW files, and auto-navigation (autonav) parameters.

C. Transaction Request Files

CFDP products required a Transaction Request File (TRF) to complete the uplink to the spacecraft. The TRFs contained the details for the handling of the CFDP products. They were used to dictate where a CFDP product would be stored onboard the spacecraft and what it would be called in memory.

The TRFs were useful, as they would allow the operations team the ability to radiate the same CFDP product to different locations onboard the spacecraft without creating a new CFDP product. Only a new TRF needed to be generated to send the same file to a different location. The TRFs were used on the ground only and not physically radiated to the spacecraft.

D. Product Naming Convention

Deep Impact employed a very detailed naming convention that was used to distinguish the many different command types from one another. A twelve, and sometimes thirteen, character string was attached to the beginning of each command and file name. This prefix was known as a "license plate." When implemented correctly and fully understood by the user, the license plate was an effective descriptor. However, the license plates resulted in several problems as discussed in Section VI.

III. Commanding the Spacecraft

A. Commanding the Flyby Vehicle

The uplink of SCMFs to the Flyby was performed in the traditional manner of someone queuing and radiating the commands. The CFDP products were uplinked, like the SCMFs, uplinked directly to the Flyby vehicle.

B. Commanding the Impactor Vehicle

Commanding the Impactor was a tremendous challenge. All commands and files that were needed by the Impactor had to first be radiated to the Flyby. Once these commands and files were received by the Flyby, additional commands were required to send them to the Impactor and then activate them if needed. Up to six products were needed to execute one sequence onboard the Impactor.

C. Uplink Product Statistics

Table 1 contains the number of flight uplink products generated and the number of products radiated for Deep Impact. This listing only includes the files that were generated and stored in the flight directories and not any products used only for Operational Readiness Tests (ORTs). Several thousand commands and files were created for the ORTs. It should be noted that the FSW team was responsible for generating their products for the FSW loads.

Table 1 - Deep Impact Radiation Statistics

Products Built for Flight		Radiation Statistics	
Product Type		Product Type	Number Radiated
SCMF	3988	SCMF	4924
CFDP File	3125		
TRF	7631	CFDP File	3129
FSW File	810		
Total	15554	Total	8053

It is difficult to comprehend how intense the operations for Deep Impact was without comparing it other missions. Table 2 contains a summary of the number of files uplinked to various other JPL missions. Deep Impact required far more commanding that the other missions listed. Even when the most intense periods of commanding the spacecraft are isolated, Deep Impact is still a much more command intensive mission.

Table 2 - Radiation Comparison

Spacecraft	Duration (days)	Total Commands and Files Radiated	Commands and Files Radiated per Day	Launch Date	Comments
Deep Impact	176	8053	45.76	January 15, 2005	Launch through Encounter
MGS	TBD	8000	TBD	November 7, 1996	Launch through March 14, 2006
	TBD	TBD	TBD		Launch through MOI - September 12, 1997
Mars Odyssey	TBD	TBD	TBD	April 7, 2001	Launch through March 14, 2006
	TBD	TBD	TBD		Launch through MOI - October 24, 2001
MRO	TBD	TBD	TBD	August 12, 2005	Launch through MOI - March 10, 2006
Spitzer	98	1855	18.93	August 25, 2003	Launch through IOC/SV - Dec 1, 2003
	TBD	4819	TBD		Launch through March 14, 2006

*This includes 4924 SCMFs and 3129 CFDP Files

D. Batch Mode

As illustrated above, Deep Impact required an enormous volume of commands and files for operations. The time required to queue and verify each command and file to radiate to the spacecraft proved to be very tedious for the spacecraft team. Activities such as a FSW load or an autonav demonstration contained several hundred files to uplink to the spacecraft. The uplink for the products used for the encounter with Tempel-1 contained well over 400 files. The solution to this problem was to develop a “batch mode” of uplinking files to the spacecraft. The development of the rad lists and software used to facilitate the process are discussed in Sections IV and V.

IV. Uplink Process Overview

Presented below are the steps that were taken to prepare products for radiation to Deep Impact.

1. Activity Kickoff

The kickoff for each activity was a meeting that was led by the activity lead. The kickoff contained a summary of the activity and relevant dates throughout the development cycle such as when inputs would be given to the MPST, subsystem review, and the Command Approval Meeting (CAM). Kickoffs were held primarily towards the end of the mission when the intensity and the increased number of activities necessitated extra coordination to keep teams focused on all upcoming activities.

2. *Delivery to MPST*

When uplink products were needed the activity lead would notify MPST. The deliveries were repeated through the entire uplink process whenever a problem was discovered.

3. *Uplink Product Generation*

The MPST would then take the delivery from the activity leads and generate the uplink products. The uplink products were reviewed for sanity by the MPST and then stored so the spacecraft team could access them.

4. *Products to Testbed*

Once the activity leads had the uplink products they were able to simulate the spacecraft activities using the Deep Impact testbed. The results of the testbed run would later be discussed in the sequence approval meeting (see below). If a problem was found within the testbed data then the delivery to the MPST would be repeated and the products would be retested.

5. *Timeline Delivery to MPST*

The delivery of the timeline would occur with delivery of inputs to MPST, in concurrence with the testbed run, or shortly after the testbed run was completed. Ideally the timeline would arrive before the testbed run so the Predicted Events File would be complete around the same time as the testbed run providing the subsystems with all of their review inputs. However, the timeline could not always be finalized before the testbed run so MPST would be forced to delay the release of the Predicted Events File to the subsystems.

6. *Predicted Events File*

The Predicted Events Files, or PEF, is a file that contains all of the expanded sequences and constraint checking. The PEF predicts the ground and spacecraft events in a form that readable by the user with some training. The PEFs were used to check for flight rule violations and by the subsystems to review each major activity that was planned to take place on Deep Impact.

7. *DSN Keyword File*

The DSN Keyword File (DKF) is generated from the PEF and is used by the Deep Space Network (DSN) to properly configure the antennas for each pass. DKFs are important for all missions to ensure that all data is captured on the ground and that the spacecraft be commanded. This was especially important for Deep Impact since so many activities relied on real-time activity. If a DSN station had a problem and Deep Impact missed a pass the entire activity was in danger of not being able to execute.

8. *Subsystem Review*

Once the modeled PEF and testbed data was available the subsystems would review the activity. If any issues were found then the activity lead would redeliver inputs to the MPST and the process would repeat itself. Redeliveries would continue until the subsystems were satisfied that the objectives of the activity were met and that the health and safety of the spacecraft was maintained.

9. *Sequence Approval Meeting*

The sequence approval meeting was the official forum for subsystems and activity leads to discuss the activity. Each subsystem that was involved in the activity would approve or disapprove of the activity. Depending on the severity of the changes, a new testbed run and PEF could be required with an additional subsystem review.

10. *Uplink Summary Generation*

Uplink summaries (ULS) were generated by the MPST using the timeline as an input from the activity lead. Uplink summaries would contain all real time commands and CFDP/TRF pairs that would get uplinked to the spacecraft for an activity. The summaries would also contain information unique to each command or file that was used to verify that the proper command was uplinked to the spacecraft. The uplink summaries contained anywhere from one item to 469 items.

Unlike other MGSS missions, Deep Impact did not include uplink windows on the ULS. Initially, this was thought to be a possible risk but the mission proved that they were not needed. This may have been a result of Deep Impact using near continuous DSN coverage but it may also be an area future missions can use to eliminate workload.

11. *CFDP Radiation List*

Typically, only activities that contained a large number of uplink products (greater than 25) or that were time critical (such as an ephemeris update) to uplink received rad lists. Activities would contain anywhere from one rad list to eighteen rad lists.

12. MPST Review

Once the ULS and rad list were complete the MPST would review the contents of each item. This involved using the timeline to verify the contents of the ULS and rad list and also verifying that the commands existed and could be accessed by the command system. Once the review was complete, the ULS and CFDP Rad List were taken to the Command Approval Meeting. No products were uplinked to the spacecraft without the MPST first reviewing the uplink summaries and rad lists.

13. Command Approval Meeting

The command approval meeting was where the ULS and rad lists were approved for radiation to the spacecraft. These meetings would consist of a walkthrough of the ULS and provide participants with one final opportunity to voice any concerns they might have about an activity. All major activities had a command approval meeting before the products were uplinked to the spacecraft.

14. Uplink to the Spacecraft

At the conclusion of the command approval meeting, the uplink summary and rad lists were taken to the engineer who was responsible for uplinking the commands and files to Deep Impact. All commands and files were sent to the Flyby vehicle and later transferred to the Impactor if needed.

V. MPST Toolbox

The uplink process described in Section IV would not have been possible without the tools developed in flight by the MPST. The following toolbox contains tools that were developed for operations. The tools were constantly modified during flight as the mission's needs were defined. The final changes were implemented six days before the encounter with Tempel-1.

The MPST tools not only permitted mission operations to function as they did, but the MPST was actually able to decrease staffing even as the number of activities on the spacecraft increased.

A. Uplink Product Generation - *gen_di_cmd_pro*

Shortly before launch, the naming convention for commands was changed to the license plate method described in Section II. When this change was implemented, Deep Impact could no longer rely on the Automated Sequence Processor (ASP) to generate and store uplink products. Traditionally, spacecraft team members use the ASP to generate their own uplink products. The ASP is programmed to assign generic names to uplink products and Deep Impact chose instead to use the detailed license plate naming convention. As a result of that decision, the ASP was eliminated from operations.

Without the ASP, the MSPT was forced to manually generate and store the uplink products for the spacecraft. Since the ASP was originally scheduled to handle this workload a new tool needed to act as the ASP and be smart enough to utilize the new naming convention.

The tool to satisfy these requirements was *gen_di_cmd_pro*. This Cshell script functioned just as the ASP would have with the exception that the MPST was responsible for all uplink products being produced instead of the spacecraft team. The *gen_di_cmd_pro* tool was responsible for producing all uplink products except the autonav parameters and the FSW files and FSW TRFs.

The *gen_di_cmd_pro* tool successfully replaced the ASP and as an added feature was able to operate in a batch mode using a text file as an input. The tool performed flawlessly throughout the entire prime mission and enabled the MPST to generate uplink products quickly and efficiently.

B. Modeled Products Tool - *gen_di_mdl_pro*

As previously mention, the MPST had the responsibility of generating PEFs for the subsystems to review. To accurately model an activity on the spacecraft, all commands that will execute must be included in the PEF. These commands come from both real-time commands and the CFDP products.

Deep Impact PEFs were particularly challenging. One challenge was due to the large number of command and sequence files that were required for modeling. The best way to illustrate this is to compare how many files containing commands are needed to produce a PEF for other missions that are similarly sized to Deep Impact. Mars Odyssey and Mars Global Surveyor consistently contain less than ten files. The Spitzer Space Telescope may contain upwards of 50 files but these files are all contained in one delivery so they are easily included in the PEF.

Deep Impact PEFs could contain nearly 400 files for a given week. These products could include any uplink product generated throughout the mission and not bundled together like a Spitzer delivery. Since commands were used repeatedly, the majority of the commands and files used on Deep Impact were built as relative-timed commands and sequences as opposed to absolute timed products.

Relative-timed products presented another challenge when building the PEFs. The times within the relative-timed files must be manually edited to reflect when the command will most likely execute on the spacecraft. Complicating the process even more is the fact that the same file could be used several times throughout a PEF. The *gen_di_mdl_pro* script was developed to assist with the PEF generation. The primary function of *gen_di_mdl_pro* was to replace the generic times within relative-timed commands and files with the time the command would execute on the spacecraft. Some files would contain hundreds of commands requiring that hundreds of times be modified so this was an important task to have automated.

C. Uplink Summary, Radiation List, and Utility Tool - *spider*

Spider was a tool that evolved throughout the entire mission. The initial requirement for *spider* was to verify that the commands on the ULS were accessible to the command system. The next revision of *spider* was used to assist the MPST with the review of the ULS. *Spider* was locating several ULS errors early in the mission. The errors were not a result of mistakes made by the user but bugs and inefficiencies within the current software that was provided to MPST to facilitate ULS generation. There was not enough time for a redelivery of the software so *spider* was adapted to generate the ULS. Furthermore, the original software was not programmed to properly add Impactor commands and files to the ULS, only those whose final destination was the Flyby vehicle.

Spider was further adapted as needed when high-risk areas were identified throughout the mission to perform checks and balances. A primarily sanity check was verifying that the license plate matched the content of the command or file.

Besides ULS generation and error checking *spider* served one more very important purpose. Once the need for the rad lists materialized it was a natural adaptation for *spider* to produce the lists. The MPST worked with the CMD system engineers to produce the requirements for the list and the new CMD System. The rad lists and CMD system were developed in parallel and integrated into mission operations in April 2005.

VI. Lessons Learned

The following lessons learned from Deep Impact's daily uplink operations are what the MPST feel should be shared so that future missions can benefit from the experience gained during Deep Impact. This is not a mission comprehensive list or representative of lessons learned from other spacecraft team members, just the MPST.

1. Activity Lead Deliveries

The MPST had to coordinate the preparation of uplink products with twelve different activity leads. Early in the mission there was not an established procedure for deliveries from the activity leads to the MPST. This created a great deal of confusion shortly after the launch of the spacecraft. The problem was somewhat rectified by standardizing the timeline format but this was not strictly enforced so there were still some inconsistencies. If the activity lead concept is to be utilized in future missions their final deliveries need to correspond with consistent processes and output files. Scripting is extremely important during operations as automation saves time and eliminates human error. *Scripting cannot be used when deliveries have no consistency.*

Along with inconsistent timeline deliveries, the MPST was also delivered bad SASFs on a consistent basis. This problem was more noticeable when an activity lead first started to generate SASFs. To their credit, as they gained experience building the SASFs the number of bad files decreased drastically. The solution to this problem would be to offer some type of command training. This training would present a strategy to building commands and also show common mistakes individuals make while generating the SASFs. One to two hours would be adequate time for such an activity. The time spent in a command school would be significantly less than the time it takes the activity leads to regenerate and deliver to the MPST and for the MPST to troubleshoot the errors that come with bad deliveries.

Lesson: It is very important that written agreements detailing how one team delivers to another are developed. This will more easily allow automation to be integrated into procedures. Scripting can rely heavily on techniques such as "pattern matching" and that technique is only effective when the input is in a consistent format.

2. Real-time Command Modeling

The difficulties brought upon by the large number of relative timed real-time commands required for PEFs has already been addressed by the sequence software developers. This was a known problem before Deep Impact and Deep Impact demonstrated that this future modification should be a priority.

Lession: A more efficient method of modeling real-time commands is needed. This issue has already been addressed and future missions will surely benefit from the modification.

3. Naming Convention for Commands and Files

Deep Impact used a very descriptive naming convention for the uplink products. Reading the name of the command or file could usually tell a spacecraft team member exactly what the command or file did and where if would be located onboard the spacecraft. The problem with relying on the name of a command or file for that information is that it is easy to neglect looking within the command or file itself for that information. If the license plate was not correct then a bad command could be radiated to the spacecraft. This actually happened on multiple occasions but fortunately for Deep Impact the commands did nothing to jeopardize the heath or safety of the spacecraft. Eventually checks to ensure that the license plate and file content were the same were implemented into both *gen_di_cmd_pro* and *spider*. The MPST or the activity leads should have integrated these checks into the development process before the spacecraft launched. A descriptive naming convention is not reliable until the proper checks and balances are in place for the development of the commands and files along with a "bullet-proof" method to define the name.

Lesson: Care should be taken when using a descriptive filename so that the contents of the file are examined and not the filename. Checks can also be developed during the generation of the commands and files to verify that the name matches the contents of the file.

4. Replacing Old Processes with New Processes

The ASP for Deep Impact was delivered to the project as a means to build uplink products. The project chose to bypass this established process and instead rely on the MPST to generate the uplink products. This decision led to an increased workload for the MPST. Also, replacing an automated process with a human process introduces risk. If a command is needed quickly for a mission critical event, an MPST member must be available to build a command. When commands are needed during overnight operations, a MPST member would need to support the activity by working overnight or by receiving what turned into frequent wake-up calls.

Lesson: When an established, automated system is available, every effort should be made to take advantage of that system. Replacing an established system with a newly developed system is risky and time consuming.

5. Uplink Product Development Cycle

Deep Impact suffered from the combination of a complicated spacecraft and a short journey from Earth to comet Tempel-1. The result of this is that activities were put on an accelerated design schedule and occurred several times per week resulting in several meetings late in the day and during most weekends. This did not allow for any type of consistent development cycle to be established. The benefits of a consistent development cycle are that team members have a better grasp of when activities need to be reviewed and when the review meetings are held. Deep Impact was a victim of circumstance and the mission was too hectic for a consistent development cycle. This demonstrated how important a consistent development cycle is.

Lesson: A consistent development cycle is extremely important for operations. The entire process is more efficient and safer when teams are in the habit of performing for a consistent schedule. This will also eliminate the need for meetings late in the workday and on weekend that can decrease team morale.

6. Pre-launch Uplink Process and System Characterization

Several areas of the uplink process were not characterized until the spacecraft had already launched. The first of these is that the Impactor command names were not defined until several weeks into operations. As mentioned previously, the Impactor required several products just to activate one onboard sequence. The process of developing these products needed to be exercised before flight. The Impactor naming convention was extremely complicated and resulted in frequent erroneous deliveries to the MPST. A tool needed to be developed and made readily available to all activity leads before launch to automate the production of the Impactor commands.

The Impactor commands were only a small contribution to the overall command volume required to operate Deep Impact. Activities would routinely require hundred on individual commands and files to uplink to the spacecraft. The uplink sessions were time consuming and put a tremendous strain on the spacecraft team. The "batch mode" was a huge time and sanity saver but it was not available until the mission was over half way complete. This problem should have been addressed pre-launch and rectified before the spacecraft was launched.

Lesson: Deep Impact needed to have all elements of its uplink process defined before flight. Many missions can get away without defining all processes if they have a long cruise period such as Cassini's voyage to the Saturn system. Deep Impact did not have that luxury and if would have been much safer and efficient to have all procedures defined and practiced pre-launch.

7. Background Sequence and Mini-Sequence Development

Background sequences are used by spacecraft when performing routine spacecraft operations, such as FSW diagnostics and communicating with the DSN. Deep Impact did not initially use background sequences for mission operations but began doing so on February 9, 2005 for the fifth week of operations. Before utilizing the background sequence, each individual activity had the responsibility of including the background sequence content within the activity or there would be a small stand-alone activity to accomplish the task. This resulted in many extra ULS and commands to build. The background sequences eliminated this issue.

Background sequences are not the only method that could have increased efficiency. The activities that took place on Deep Impact were uplinked to the spacecraft as individual commands and multiple sequences and tables. This required hundreds of commands and files to be uplinked in a single session for one activity. A more efficient means of achieving this would have been to use absolute timed mini-sequences to conduct the activities onboard the spacecraft.

Lesson: If a new uplink capability such as CFDP is to be utilized, it should not be at the expense of established methods of commanding a spacecraft. If the traditional method of background and mini-sequence SCMFs were used then the number of uplink products would have drastically been reduced,

VII. Conclusions

Deep Impact was a successful mission despite the many challenges the spacecraft team was faced with just to command the spacecraft. Future missions need to take caution when implementing new technology and ideas so that they avoid reinventing processes that have proven reliable to past missions. CFDP and the new naming convention were examples of ideas that were successfully demonstrated in flight but drastically increased the workload. The MPST tools eventually molded a process together that allowed the less efficient methods to function on a daily basis. The MPST tools were so effective that they were able to decrease staffing as the mission progressed and became more intense.

Deep Impact was fortunate enough to have a team of capable and dedicated engineers who, at a great deal of personal sacrifice, were able to pull together to ensure that the encounter with comet Tempel-1 was a success.

Acknowledgments

Complete for the final version of the paper