CONCURRENT ENGINEERING: SPACECRAFT AND MISSION OPERATIONS SYSTEM DESIGN

J. A. Landshof*, R. J. Harvey*, and M. H. Marshall**

The Johns Hopkins University
Applied Physics Laboratory
Laurel, Maryland 20723-6099

Abstract

Despite our awareness of the mission design process, spacecraft historically have been designed and developed by one team and then turned over as a system to the Mission Operations organization to operate on-orbit. By applying concurrent engineering techniques and envisioning operability as an essential characteristic of spacecraft design, tradeoffs can be made in the overall mission design to minimize mission lifetime cost. Lessons learned from previous spacecraft missions will be described, as well as the implementation of concurrent mission operations and spacecraft engineering for the Near Earth Asteroid Rendezvous (NEAR) program.

Introduction

The traditional approach of system development (requirement definition, specification development, preliminary and detailed design, fabrication, and test) is a long, cumbersome, and frequently costly process. Current system engineering techniques for system development such as concurrent engineering and rapid prototyping can be much faster, and, consequently, cheaper. There may be increased risk in this approach, however, the benefits generally outweigh these risks. In cost and schedule constrained programs such as Discovery programs, higher risk must be tolerated to achieve the goals of faster, better, and cheaper.

Concurrent engineering is defined here as the simultaneous development of two or more interacting systems from the earliest stages of the system life cycle through the design and development process. System engineering includes in part the allocation of system requirements to subsystems, and when two or more subsystems' requirements overlap, or when a system-level requirement could be handled by two or more subsystems, concurrent engineering techniques can be used to arrive at an optimal solution. This paper will describe what is meant by concurrent engineering as it applies to the development of space systems, focussing on the concurrent design and development of a spacecraft and the mission operations system that will be used to operate it on orbit. The benefits and costs of concurrent engineering in this application will be discussed, and concurrent engineering methods will be presented. Then, specific examples of lessons learned from past space system development programs at the Johns Hopkins University Applied Physics Laboratory (JHU/APL, or APL) will be presented, along with a work-in-progress snapshot of concurrent engineering in practice on the Near Earth Asteroid Rendezvous (NEAR) mission .

Concurrent Engineering of Space Systems

A space system includes a spacecraft and the systems with which the spacecraft will be operated once it is in space (the mission operations system, or MOS). At the very start of a mission, requirements are allocated between the spacecraft and the MOS (e.g., existing MOS infrastructure may require a certain frequency for uplink and/or downlink, requiring the spacecraft telemetry system to be built in compliance thereof), ideally by a mission system engineer. After these top-level allocations are made, requirements in both systems are further allocated to subsystems within each, by the cognizant system engineers. Even when requirements are allocated along clear lines, simple decisions in one system can have

* Member, Senior Professional Staff
** Member, Principal Professional Staff

great affects on the other. A mission system engineer is crucial to resolve conflicts, and to make decisions as to what requirements should be done where.

As the spacecraft and the MOS are being designed, constant communication between the two development activities is crucial in order to end with a space system that works well as a whole. Therefore, the communications between spacecraft subsystem design efforts and MOS design efforts must include design decisions as they are being made. Communication must occur at the lowest possible level, between individual engineers responsible for subsystem design if possible. In some cases, relatively minor changes in spacecraft or instrument design can significantly save in operations costs. For example, thermal and power robustness may eliminate the need for complex analysis of every maneuver sequence, saving time and money in the development of sequence uploads.

A mission level system engineer should be designated at the start of a program by the program office, with the capability and responsibility to perform requirement tradeoffs at a high level. Too frequently, all flexibility and operability is pushed onto the ground system and mission operations functions to save development costs in the spacecraft. This is often the correct approach (complexity versus reliability tradeoffs in the spacecraft can be prohibitive), however, in the current budget environment, this is not always the optimal approach.

Benefits

There are many benefits to designing major elements of space systems concurrently. Concurrent engineering allows optimal systems solutions across disciplinary boundaries, with the added bonus of often doing so in less time at a lower cost. One inevitable outcome is the education of engineers about each other's systems. In the case of spacecraft subsystems and mission operations, the better mission operations understands the spacecraft, the more safe, efficient, and reliable mission operations is going to be. The better trained and educated the mission operations team is the better they will be able to respond quickly and correctly to solve any anomaly that might arise on the spacecraft.

In the development arena, concurrent engineering can allow for more flexible response to changes in requirements. If a requirements change is forced on the spacecraft late in the design cycle, it is often very costly to modify flight designs. If the MOS is able to respond to the requirements change, costly delays in the spacecraft development program are often avoided, albeit at some potential expense to the MOS development effort.

Finally, if a spacecraft is designed from the outset with operability in mind, fewer people may be required to operate it. Since personnel are usually the driver for mission operations post-launch costs, lowering the number of personnel required to operate a spacecraft can dramatically reduce mission operations', and thus the overall program's, costs.

Costs

Concurrent engineering does not come without costs. There is often an increase in the time required for communications between development groups. This is especially true early in the program, during conceptual and preliminary design phases when teams may be small and design time precious. During the system development period, subsystem teams can not just build their box in isolation. They must continue to work with other elements as designs are solidified, to ensure a working system at the end.

Finally, perhaps the most critical time consuming effort is in convincing all team members that concurrent engineering is a worthwhile effort. Concurrent engineering runs counter to traditional subsystem development processes. Often, concurrent engineering can seem to overstep 'turf,' when for instance a mission operations person requests changes in the command system design. A strong mission systems engineer can smooth the turf battles, but it is time consuming. Once everyone realizes that the true end product is the space system, not a subsystem, these concerns tend to go away.

Methods

Two methods are currently being used at APL by the mission operations organization in conjunction with spacecraft development programs to enable the concurrent engineering process for space system development. These methods are the identification of a spacecraft specialist in the early prelaunch phase, and the development of the spacecraft ground system ICD.

The Spacecraft Specialist

The spacecraft specialist is responsible for providing the bridge between mission operations and the spacecraft development team. This person should ideally have both a spacecraft hardware and an operations background.

During the initial phases of the mission, the spacecraft specialist job is to work with the spacecraft system engineer, and subsystem designers, to ensure operability is a consideration in all design phases. It was found on previous programs that just asking subsystem designers to think about operations did not work -- someone was needed, paid for by mission operations, whose job was to look over the shoulders and comment on designs as they evolved.

Early on, as mentioned above, some subsystem designers felt that mission operations was intruding into their territory. As time went on, though, almost all came to understand and appreciate, and in some cases even demand, the perspective brought to the table by the spacecraft specialist. Critical to this success, however, is the credibility of the spacecraft specialist.

The Spacecraft/Ground System ICD

One of the primary products of the spacecraft specialist in the early program phases is the Spacecraft/Ground System Interface Control Document (ICD). This document captures the interface between the ground, both the GSS and the MOS, and the spacecraft, and should be completed before the spacecraft Critical Design Review (CDR). For each spacecraft system, and subsystem, the ICD defines commands, telemetry, and operating rules, as they are known at that point in the program. With this document in hand, the ground system development team can proceed to build the command and telemetry processing system, and the spacecraftdevelopment team can proceed with the development, integration and test of their subsystems. Communication between the teams is still required, though; the ICD is the beginning of the process, not the end.

Like most documents, the Spacecraft/Ground Systems ICD is most useful during its development, not by its use. Requiring that both mission operations and the spacecraft subsystem personnel think about command formats, telemetry, and operating rules very early, in order to develop the ICD, is the very essence of concurrent engineering.

Space System Development: Past Experience

Over the years, the Applied Physics Laboratory has built over fifty spacecraft. Virtually all of these were one-of-a-kind spacecraft built for a specific research purpose. With this history comes an institutional way of doing business. Programs have tended in the past to be very focused on the spacecraft. As current missions have required more of a mission focus, the institutional ways of doing business are changing. On previous missions, there were a number of areas where concurrent engineering might have reduced the cost of mission operations development and implementation, helping to reduce overall mission costs. Areas of spacecraft design where mission operations' input early on might have proven beneficial include spacecraft commanding, telemetry, onboard memory management, onboard data processing, and the testing and testability of some subsystems. Examples are given below of specific lessons learned on recent APL space system development programs. In some cases these examples refer to the standardization of designs throughout the spacecraft, while others refer to particular design change recommendations to make operations more efficient.

Commanding

Mission operations' sole connection to the spacecraft after launch is through the command and telemetry links. The only path for mission operations to affect anything on the spacecraft is via commands from the ground. In the early days of space, spacecraft were launched with fixed timelines of activities; no changes from the ground could be made. Now, of course, spacecraft are built to respond to ground commands to carry out activities. The development of the commands to be sent to the spacecraft is, in fact, the primary focus of mission operations today. Therefore, designing the command interface to the spacecraft with operability offers perhaps the best opportunities for a more easily operable spacecraft, which in turn can reduce the size of mission operations considerably. One particular area of interest is in the types and formats of the commands themselves.

A standard command format being mandated throughout the spacecraft would enable the mission operations team to develop a standard mechanism for the automated generation of commands. Hard-coded workarounds in flight software that require special command types not only escalate the cost of development, but reduce the speed of an automated command generation process. A standard command format should be applied to serial data commands, which might include an "opcode" at the start of the data to indicate the command type or functionality. Mode change commands should not be of the type where each bit addresses some particular function; to change a single element with such a system, each bit must be respecified to its current state. This is a nightmare for mission operations! As an example, one program had a command design where four bits of a serial data command represented the enabling and disabling of four different data formatters. Every time one particular formatter was to be enabled, the previous state of the others had to be known. If the wrong state had been assumed, the command may have inadvertently disabled one formatter that should have remained enabled. This could have caused something as critical as communication of spacecraft housekeeping data suddenly being lost when science data was enabled for on-board recording. If the function of controlling each formatter had been made a separate "opcode," each formatter could have been controlled individually without having known each other's previously commanded state. The creation of the command loads would have been easier, the checking of those commands loads more reliable, and mission operations workload reduced significantly.

Standard command formats also may reduce mission operations development costs by making spacecraft state determination and tracking easier. Lower fidelity models of the onboard processor, its memory, and its state would still provide all necessary functions, but require less design, development, and maintenance, thereby reducing costs. Automated command generation schemes also can reduce personnel requirements.

Telemetry

To assist in the area of spacecraft control and performance assessment, every command, whether executed in real-time or delayed, must have telemetry which allows for the verification of proper execution or rejection. For serial data commands some means of verification are required (at a minimum the data should be reflected back into telemetry). This is essential in determining that a command was not only correctly received by the command system and transmitted from the command system to an onboard subsystem, but was in fact properly executed by the intended subsystem.

Tracking what the spacecraft has done since the last contact with the ground is very important for mission operations. To support this requirement, the spacecraft should have a command history buffer. The size of this buffer should be changeable by uplink command. Stored commands and commands from macros should be logged, but not necessarily data loads. Downlink of the buffer may be by ground command, to conserve downlink bandwidth. This history buffer capability allows for the assessment that a command was rejected for reasons other than not being properly transmitted from the command system. This allows

mission operations to assess spacecraft health more easily and quickly, an important factor especially on low earth orbiters with short ground contact durations.

Memory Management

Other means of standardization include the uplinking and downlinking of on-board processor's memory locations or specifically, the use of data structures. Data structures allow the loading of a processor's memory without knowing the exact locations, which could change should the processor's code be re-linked. The functionality of the processor's software should allow for the uploading and downloading of these data structures by a specific ID number. On a recent mission this capabiltity was not built into the flight software, requiring the downlinking all of the data structures at once as opposed to each data structure individually by ID. This created the requirement for additional ground software that would search through the entire downlink and find the particular one of interest.

Onboard Data Processing

On spacecraft where housekeeping data is not continuously recorded, there should be a capability for a buffer which allows the routine periodic sampling and storage of critical parameters. Most likely, throughout the life of a spacecraft's mission, different parameters will vary in their criticality. Therefore, the capability should exist for allowing ground commands to change which parameters are sampled and their periodicity. The buffer obviously must have a particular size limitation, so in cases where data will be lost because it cannot be downlinked for long periods of time, it would be highly desirable from a mission operations assessment perspective to be able to download this data to the on-board recorder for later retrieval. On a previous spacecraft a similar type buffer was limited to the sampling of certain unchangeable parameters and its capacity allowed for up to 5 orbits of sampling at a rate of one sample per 200 seconds. The rate was changeable; however, as the sampling rate was increased, the amount of time between required downlinks was reduced. In these cases, it would have been advantageous to have the capability of transferring it to the on-board recorder. Also, as the mission progressed, certain parameters which were "hard-coded" became invalid. In these cases it would have been beneficial to replace those with other critical parameters.

Such a capability would give mission operations insight into spacecraft state between contacts and allow performance assessment and trending of critical parameters as they vary throughout a mission.

Testing

Also in the area of performance assessment, for any processor or recorder (either solid state or tape), there should be a method of loading a standard data test pattern in each processor or on each tape such that it may be downlinked through telemetry and run through a bit-by-bit comparison to a ground image of the same pattern to certify memory validity and periodically measure bit error rates.

Summary of Lessons Learned

In all of these cases, if the Mission Operations Team was involved in the specification of spacecraft design requirements, the overall mission operations cost would have been reduced through both a lowering of system development costs and an increase in efficiency in the performance of mission planning, control, and assessment tasks.

Use of Concurrent Engineering on the NEAR Mission

The Near Earth Asteroid Rendezvous (NEAR) program was officially turned on in December of 1993. Prior to that, a small study team had been working on the conceptual design of the mission and the spacecraft. In August of 1993, mission operations was asked to provide input as to spacecraft design considerations for the NEAR mission which would enhance operability. Below, the input provided for spacecraft design features are listed, and the current status of each is described. Following that, other activities highlighting the use of concurrent engineering on NEAR are described.

It must be strongly emphasized that the NEAR space system is still being designed; as of the writing of this paper (July 1994) both the spacecraft and the mission operations system are in the design and develop-

ment stages. What follows is a snapshot of work-in-progress; by the time of the SpaceOps '94 symposium, (November 1994) the spacecraft will have passed its critical design review, and the presentation for this paper will update the following material.

Mission Operations Inputs for NEAR Spacecraft Design

The following items (numbered) were listed by mission operations in August of 1993 as design considerations for the NEAR spacecraft, and can be seen to come from the experiences described above. They are listed in no particular order:

1. "Spacecraft and RF system must have power/thermal margin to transmit continuously for 8-hour contact. If a contact is delayed, actual transmission time may be longer"

Current Status: The NEAR spacecraft can transmit continuously during all mission phases.

2. "The spacecraft will have a data summary area in the command and data handling (C&DH)system computer. The 'Data Summary' requirements include:

– At least 5 data points for each important telemetry parameter (high, low, average, time of high, time of low)

– A variable length 'Anomaly Data' area where data triggered/ written by the autonomy system is stored."

Current Status: The NEAR spacecraft C&DH software requirements specification includes all of the above requirements.

3. "The NEAR Solid state recorder (SSR) memory must be non-volatile."

Current Status: The NEAR spacecraft solid state recorder memory is volatile - shutting off the power causes the data to be lost. However, the power should never have to be turned off, so this is not a seen as a critical issue by mission operations. The SSR is a purchased component, with an existing design, and designing a new recorder would have been cost and schedule prohibitive.

4. "Realtime telemetry must be available to the SSR and telemetry system simultaneously."

Current Status: The NEAR spacecraft can both record and downlink housekeeping ('realtime') data simultaneously. This feature can be used to prevent the loss of spacecraft housekeeping data in the event of a transmission error.

5. ."The SSR must have random access capability. Downlink of selected time periods of selected parameters is required."

Current Status: The NEAR solid state recorder has some capability for random access, but not by time and parameter. The ground will have to model data recording functions in order to know what particular SSR memory addresses to downlink for particular data. The onboard data rates of all instruments and subsystems are controlled by ground command, so this is not seen as a problem.

The following items concern the onboard spacecraft telemetry processing and anomaly detection and correction processes, collectively known as autonomy

6. "Autonomy rules should include chaining (i. e. if A is true, then check if B is true, then take some action) and arithmetic (i.e. allow the multiplication of a voltage and current telemetry parameters to check on power consumption."

Current Status: The onboard autonomy does not allow for arithmetical functions on telemetry, but it does allow for limited logical checks (ANDs and ORs of particular telemetry values). Mission operations and the flight software team are still negotiating this requirement.

7. "Autonomy should have access to SSR (to support onboard trending in case of fault detection)."

Current Status: This has not been designed into the system.

8. "Autonomy should be able to write data and conclusions to an'Anomaly Data' area of the 'Data Summary'."

Current Status: The onboard processor will capture that information which caused a

particular autonomy rule to be triggered. The data will be stored in a known location, and can be downlinked.

9. "To minimize commanding, a data region accessible to commands is necessary."

Current Status: The intent here was to reduce the amount of commanding required by allowing mission operations to uplink changes in data for previously transmitted commands. As the design matured, mission operations and the software team agreed on a scheme utilizing onboard sets of commands, called macros, invoked by a smaller set of uplinked commands. All macros are uploadable, changeable, etc., and can be used over and over again. A great deal of preparation will go into the design of the macros to ensure their reusability.

Other NEAR Concurrent Engineering Activities

Significant interaction between the spacecraft and mission operations systems design efforts is occurring in the areas of flight and ground software. Regular meetings are held, conducted by the flight software system engineer, the MOS software lead, the mission operations manager, and the cognizant technical leads for specific subsystems under discussion each meeting. Requirements are negotiated, specifications reviewed, and implementation issues aired and resolved among all the parties. Additionally, the mission operations manager has been asked to be on the review panel of the flight software preliminary design review.

Mission Operations and the spacecraft design team are working together shoulder to shoulder, in many other areas. Load management schemes, maneuver algorithm design, etc. are all being worked on together to make sure the final space system design is a good one. All teams seem to recognize the importance of strong spacecraft/operations interaction at this important stage of the NEAR mission.

Conclusions

Concurrent engineering is a technique which can work to provide a better space system, in less time, while substantially reducing total program costs. Inherent advantages of teams working together are gained, at the cost of a little more communication and flexibility. Based on our belief in the benefits of concurrent engineerirg and lessons learned from previous space missions, the NEAR mission operations team is taking an aggressive (but tactful!) approach to concurrent engineering of the spacecraft and the mission operations system. Lessons learned from past space systems development programs have given the NEAR project team a leg up, and we are using those lessons to our advantage. The NEAR project is using concurrent engineering as the basis for the system design, and both the spacecraft design team and mission operations are profiting from the close working relationship. The payoff to date is evident; we are confident that future payoffs of this approach will enable NEAR post-launch costs to be constrained to an optimal level.