Workshop on Partipation Opportunity on the Brazilian Scientific Satellite
São José dos Campos, Brazil, October 19–20, 1994

CONFLICTS IN MISSION DEVELOPMENT AND EXECUTION: NOTES FROM THE FRONT

Malcolm D. Shuster
Department of Aerospace Engineering,
Mechanics and Engineering Science
University of Florida
Gainesville, FL 32611, USA
m.shuster@ieee.org

Abstract

The various segments of mission development and execution are intimately related to one another, and extreme care must be exercised in order to optimize the overall performance of a mission. The lack of communication between different mission segments and the lack of mutual understanding between the personnel of different segments can often lead to obstacles which severely hamper the attainment of mission objectives. Examples of such conflicts are given.

Introduction

A spacecraft mission is composed of several components or segments, all of which must work together. Nowadays, in the inauguration of a space program, the interrelation of these different segments is easy to manage, because the mission goals are usually limited, the spacecraft is simple, the technology already proven in other space programs, and the people new to the job, highly motivated, flexible and enormously hard-working. The first Brazilian spacecraft, SCD-1, was such a mission, profiting from the harmonious working of many dedicated engineers and scientists at INPE.

As the Brazilian space program grows, however, it may suffer from many of the illnesses of longer established programs. In the hope of avoiding these illnesses, the present sketch is offered. Since this sketch will highlight *problems* in program integration rather than successes, it will not name names. In general, every mission has its horror stories, even the most successful. Overall, the United States space program, from which these examples have been drawn, has been resilient and successful and generally able to recover from the minor setbacks of the type described below. Most missions have been hugely successful. Yet there is value in examining some of the ways in which things go wrong.

Project Flow — An Idealized Picture

We may understand a typical spacecraft mission according to the supposed order of development. At the beginning if not the head of this development is the project scientist, who has certain scientific goals which he may wish to accomplish. These scientific goals may sometimes be more rightly called engineering goals. However, these are generally separate from the engineering which is required for the execution of the spacecraft and its support, so from the standpoint of the mission engineers, it may be considered always to be science.

The project scientist makes his desires known (often repeatedly and for many years) and, if he is persistent and lucky, funding will be made available eventually to build a mission to carry out these goals. The management of the mission, however, is not entrusted to the project scientist, who generally lacks the background for managing the program, but rather to a program manager, who has experience in managing large groups of engineers, analysts, programmers, technicians, vendors, politicians, the press, and the fiscal overseers. The program manager is God, but he is God with a budget, which may be as much as several billion dollars over many years, and he has a goal, which was set by the project scientist. He must attempt to satisfy the needs of the project scientist within the budget. In our idealized model of a project, he can accomplish his goal.

After the program manager (and his entourage of project managers) comes the hardware engineers (some of whom may be project managers within the program and may also include the program manager). The hardware engineers must select the spacecraft hardware within the guidelines determined by the program manager. These guidelines are determined by the scientific goals of the project scientist and the money allocated by the program manager. This hardware may either be the experimental payload to make the measurements desired by the project scientist, or it may be support hardware to carry out other functions of the spacecraft (power, thermal control, attitude and orbit control, attitude and orbit determination, telemetry, command processing, etc.).

Having chosen the spacecraft hardware, the ball now passes to the analytical engineers, who must develop the methods which are used for mission support. Thus, for example, attitude determination methods must be developed to determine the spacecraft attitude from its attitude sensors, or a control algorithm must be developed to generate control torques based on the attitude, and most important, methods must be developed to reduce the incredible amount of scientific data to a form usable for scientific investigation.

It is now the turn of the software engineers, who must design, develop and test the software to implement the methods developed by the analysts, and prepare users' guides (for the software that will be implemented in the ground station).

Next come the mission operators, who will receive the mission data and operate the software. Their output is the scientific data.

This data is then given to the project scientist, who will use it for his own research and distribute it to the various principal investigators.

This highly idealized picture is shown in Figure 1. Not obvious from this picture is the fact many such chains exist in parallel for each major subsystem of the spacecraft, and these subsystems may exchange information horizontally. Thus, the signal from an attitude sensor used for attitude determination may also be used directly in the control law without computing the attitude first (an

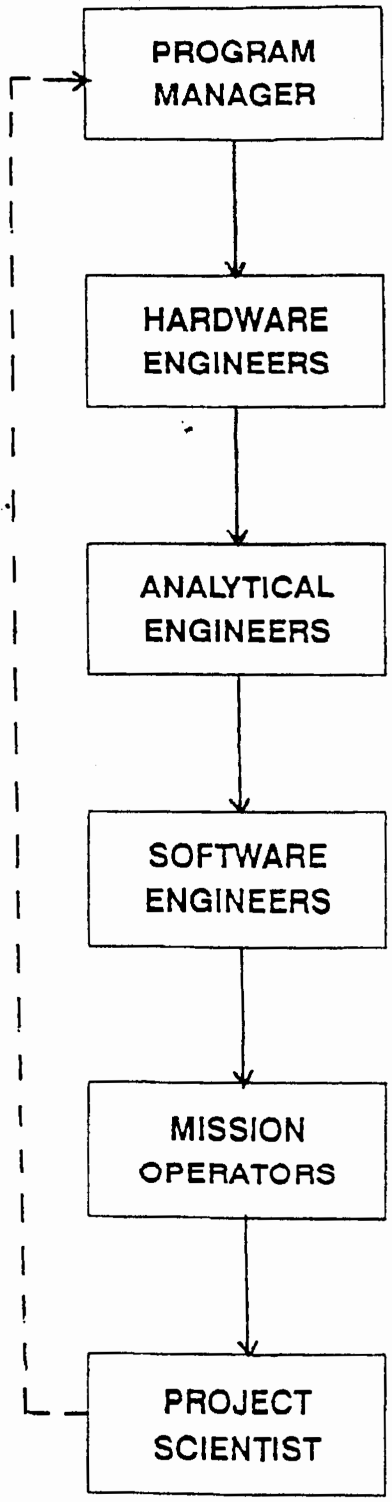

Figure 1. An Idealized Representation of Mission Development

exchange at the hardware level), or the on-board control software may use the computed attitude to compute a control signal (an exchange at the software level). We will examine in this note only problems of vertical integration of the different mission segments and we will concentrate on the attitude determination subsystem, because this is the area of expertise of the writer.

Project Flow — A More Realistic Picture

Figure 2 presents a more realistic depiction of the project flow. To some extent, the different segments are isolated from one another, seemingly make decisions without regard for the requirements of other groups along the chain of command and in some way seem to be sabotaging the mission. What actually happens, in fact, is that the different segments tend to be sensitive to the needs of their nearest neighbors (well, at least most of the time), but are totally unfamiliar with what is needed further downstream. In cases where different segments are performed by totally different organizations, the communication between nearest neighbors may break down. This is often responsible for conflicts between the hardware engineers and the analytical engineers.

Conflicts between the project scientist and the program manager are usually due to disparities in funding. The program manager cannot accomplish everything the project manager may desire within the budget constraints. This may occur because the original level of funding was unrealistic, or because the project scientist has tried to add additional experiments, or because a vendor has underestimated the cost of producing the components of the experiment. Cost overruns are quite common and have resulted in either a reduction of the original scientific goals, cost cutting elsewhere, or sometimes termination of the mission.

Examples of conflicts between hardware and analytical engineers are quite common. The hardware engineer generally does not have the capability of evaluating the sensors in a manner which is useful to the analytical. He is generally satisfied with giving a conservative 3σ value for the noise error about individual axes, while the analyst often requires an accurate measure of the full covariance matrix, a concept generally not even understood by the hardware engineer. Gyros, in particular, are often specified by the hardware engineer, even the manufacturer, in insufficient detail. The hardware engineer may decide to locate the sensors on the spacecraft in a manner which is convenient for the construction of the spacecraft but which is disastrous for attitude determination. Thus, two star trackers or a star tracker and a Sun sensor may be placed with their boresights parallel, which minimizes attitude determination accuracy (attitude determination accuracy will be maximized by making these boresights perpendicular—the difference can amount to several orders of magnitude in accuracy).

In a worst case, the hardware engineer may try to economize by having an inadequate number of sensors. For example, he may supply a spacecraft with only an horizon scanner and a two-axis Sun sensor, so that attitude cannot be determined when the spacecraft is in orbit night. The only solution to the analyst is to try to interpolate the missing angle (usually called the yaw) over the portion of the orbit for which the Sun is occulted. This is a very time-consuming procedure and can lead to large errors, since attitude dynamics are generally not amenable to accurate modeling. The better solution is to add a three-axis magnetometer or a set of three-axis gyros to the attitude hardware. The specification of inadequate or barely adequate attitude sensors has the result of shifting the burden to the later segments of the mission development and can increase the operating costs of the mission dramatically.

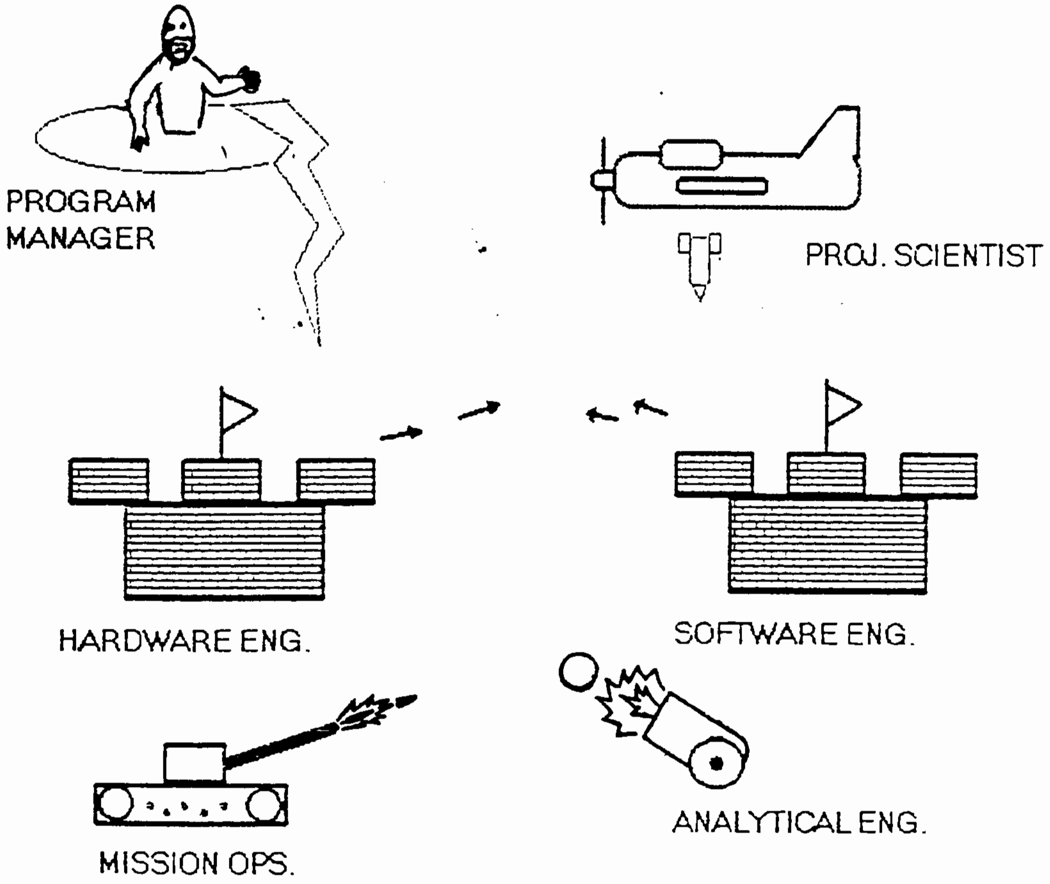

Figure 2. A More Realistic Representation of Mission Development

Sensor calibration is often one of the unappreciated disasters of the mission receiving enormous care at some stages, which is cancelled by enormous carelessness at others. The vendor (seldom) or the scientific investigator (much more frequently) may calibrate his instrument but not with respect to a repeatable attitude reference (such as an optical cube). Generally, he may not have the facilities for calibrating with respect to the cube. At integration, the hardware engineers may laboriously determine the relative alignments of different optical cubes on the spacecraft. However, if the sensors have not been calibrated with respect to the optical cube, this is of questionable value. Also, the primary reference may be a cube mounted on the spacecraft without any cubes mounted on the attitude sensors. In this case, these prelaunch alignment activities become useless for determining the attitude of the payload after launch. Also, the hardware engineers and scientists are often content to calibrate only the direction of the sensor boresight, while the orientation of the sensor about three axes must be determined.

The failure of the hardware engineers to provide adequate sensing or to carry out the prelaunch calibration and alignment activities completely and consistently forces the analyst to propose complex methodologies for attitude determination and post-launch calibration. As a general rule, the careful prelaunch alignment activities, which can take up to several months, are of very limited usefulness to the mission in these circumstances and the results are frequently discarded after launch and play little role in the post-launch analysis of the data.

The analytical engineers can also compromise the mission. The problem here is that the analytical engineer is often unfamiliar with the detailed behavior of the hardware. (Since the hardware engineer also frequently does not know how to characterize this behavior analytically, one may question whether anyone really knows the necessary information.) His tendency, therefore, is to simplify the error models and use algorithms in which the assumption of these simplified error models is key. Thus, in the early 1970's many missions tried to compute the spacecraft attitude using a Kalman filter, hoping thereby to achieve greater accuracy. The implementation of a Kalman filter requires that the dynamical model be accurate and that the sensor errors and the environmental disturbances be white. Generally, none of these conditions is satisfied, and the results for attitude determination in these unfortunate missions were disastrous; the filter would even diverge. At least one NASA spacecraft was in orbit without a working attitude ground support system for several months for just this reason. The net result was that program managers became very suspicious of the use of the Kalman filter for attitude determination (certainly with good reason), and were wary of using it again. By the early 1980's the proper implementation of a Kalman filter for attitude determination was well understood, but it was nearly a decade more before they would begin to reappear in attitude determination systems. The situation described here occurs most often because the engineers entrusted with the analytical tasks have inadequate training and simply "learn by doing." For this reason, the correct method is often regarded as "too theoretical." This is especially the case with attitude determination in which the necessary background is not a part of the standard Aerospace Engineering education.

The specification of unnecessarily complex algorithms for attitude determination generally leads to the production of cumbersome and costly software, which is often difficult to operate. In order for the mission operations staff to retain its sanity, it is necessary that the ground support software not require a great deal of human intervention. Thus, software which requires the mission operations personnel to tediously edit out individual data points, the norm for many years, imposes a heavy burden, both temporal and financial, on the mission. Mission software should be capable, therefore, of performing data validation automatically, but this aspect is frequently ignored because it increases the cost of software development (although the alternative of later operator cost is more expensive). The true value of the QUEST algorithm for attitude determination is not its speed but the fact that it validates the data point by point. Competing algorithms which occasionally engage QUEST in speed contests but ignore the problem of data validation miss the point entirely.

Inadequate operator training, typically the task of the software engineers, can also lead to disasters. More than one mission has been lost (the Soviet Phobos mission being the most recent and dramatic example) because an operator uplinked the wrong command.

And finally we must remember that the role of the analytical engineer does not end at launch. Post-launch activities include alignment determination and calibration of the sensors. To carry out these activities, one must often cause the spacecraft to execute special slews. This takes time away from scientific observation, which the project scientist is often unwilling to concede. However, without these activities, the accuracy of the data is often compromised.

To these interface problems must be added the human element. Once they leave the university, engineers are often reluctant to learn new methods (especially analytical methods) and are content to employ techniques that may be unsuited to the project at hand. In addition, there is also the resistance of a superior to accept the contention of an inferior that an error of commission or omission has been made and this resistance tends to grow exponentially as a function of the number of levels of management which separate superior and inferior. The consequences of such a situation can sometimes be tragic. The tendency nowadays seems to be to try and avoid such situations by complex and tiresome accounting procedures. Frequently, however, these only serve to provide checks on those elements of the mission design and development process which are well understood and unlikely to lead to problems and to lead to increased resistance to consider the possibility that some sources of error may reside outside the activities subject to the oversight process. The oversight process then becomes simply a legal cover against charges of negligence. Under these circumstances, the employee who discloses that the oversight process was inadequate becomes vulnerable to highly prejudicial sanctions.

An Example

As an example of how some of these problems occur in real life, we will review a mission from the late seventies, which will remain anonymous. We choose an old mission because people are less angered by criticisms of the ancient past, and also because people are more likely to have forgotten who the players were.

The mission in question was a geophysical mission employing a spacecraft in near-Earth orbit. The first problem came from the selection of the attitude sensors. The spacecraft would be supplied with only a two-axis Sun sensor and an Earth horizon scanner. Point-by-point attitude determination therefore required that the Sun always be visible, which is never the case. Thus, the yaw, the angle about the vertical, could not be determined instantaneously during orbit night, which could be as much as half of the orbit. In order to determine the yaw, one had to interpolate it from the known values on either side of the orbit. The process was extremely complicated. The inclusion of a three-axis magnetometer in the hardware, at a cost of perhaps less than $100,000 might have eliminated this problem. Instead, the spacecraft was fabricated without this sensor and the cost was pushed farther down the chain of development and operation.

The yaw interpolation algorithm was complicated to code and test and greatly increased the cost of the analytical specification of the ground system as well as the cost of the software development and validation. These two items alone probably outweighed the cost of the three-axis magnetometer. In addition, the computers of the late 1970's were very slow (the computer for this mission was, in fact, used technology entirely from the 1960's) and so processing times for attitude determination were very long. In addition, this mission would need to determine attitude at a much greater rate than did previous missions. Because of the long times required by the data processing (which also included very cumbersome data editing procedures), the mission operations personnel were required to work seven days a week, and after several weeks threatened to resign *en masse*. This never took place, but only because the spacecraft experienced a massive short-circuit after slightly more than three months in orbit.

Was this a disaster for the mission? In fact, no. A decade later one of the principal investigators noted that although the mission had only slightly more than three months of data, the termination of mission operations freed a great deal of money for data analysis, with the result that the mission was highly successful, perhaps more successful than it would have been without the anomaly (the polite word for failure). This aspect, however, was not the result of careful planning. At least, we hope it wasn't.

What Can We Do?

The simple cure for these problems is to be communicative and flexible. Unfortunately, communication takes time, especially for very complex projects, and governing bodies are reluctant to support a level of funding which allows a mission to be developed in an ideal manner. Concurrent engineering is an important buzzword nowadays. It should mean that the different components of the project are specified concurrently, so that all will fit together properly at the end. All too often it means that different components of a mission are specified and developed simultaneously without regard for the needs of the other components.

The most reliable way to avoid these problems is to keep missions small. A ten-billion-dollar mission will almost certainly face all of these problems, because the scope is so large that adequate communication is impossible. Some missions, for example, the Apollo Program, require such a level of expenditure. It is also true that the Apollo Program was not without its moment of intense acrimony. But frequently it will be better to build three spacecraft for $150 million each than a single spacecraft for $300 which carries out the same activities. The total cost is increased but so is the amount of experience gained and the likelihood of success.