Managing Requirements

by Ivy F. Hooks

Several years ago, I called upon an old acquaintance who had recently assumed management of a troubled program. I told him that I would like to help him manage his requirements. He told me that he did not need any help, because he had asked the advice of a mutual friend and NASA manager. That advice was: "Just say 'no' to all proposed changes."

The advice was not necessarily bad advice, it was just not appropriate to this manager's problem. A major problem with the program was that it had very poor requirements that could not be satisfied within budget or schedule. I have no idea what the program manager actually did, he moved again fairly soon, but the program has since been canceled.

You may be surprised to learn that you are not really managing requirements. Program managers tend to focus on subjects other than requirements. This occurs because of a bad assumption -- the manager assumes that everyone knows how to write requirements, thus the requirements process will take care of itself.

Most program managers come from a technical background and will focus on the non-technical aspects of the program, since these are new and alien. The program managers know that they do not fully understand budgets, so more attention goes to budgets. Besides, the program manager's boss will focus on budgets, and not on requirements, so the program manager places more attention on that which interests the boss.

Most people understand that bad assumptions are traps just waiting to get you, and this bad assumption -- ***requirements will take care of themselves*** -- is no different. This paper will look at how this bad assumption can wreck havoc with a program, the types of problems that occur because of this bad assumption, and what NASA program managers can do to improve their requirement management process.

How Failure to Manage Requirements Affects Programs

If the program requirements are not well understood, there is not much hope for estimating the cost of the program. In today's environment -- 15% overrun and your program may be canceled -- it is foolish to budget incorrectly. But you cannot budget correctly without a good set of requirements.

Werner Gruhl developed a history of NASA programs versus cost overruns (Figure 1). He attributed much of the problem of cost overruns to the failure to define the program properly in Phase A and B so that good cost estimates could be made.

Even with the best cost estimate, many programs will encounter overruns because of changing requirements. This phenomenon is one the aforementioned program manager was trying to avoid. The time to avoid this problem is not in Phase C or D but at the beginning of the program.
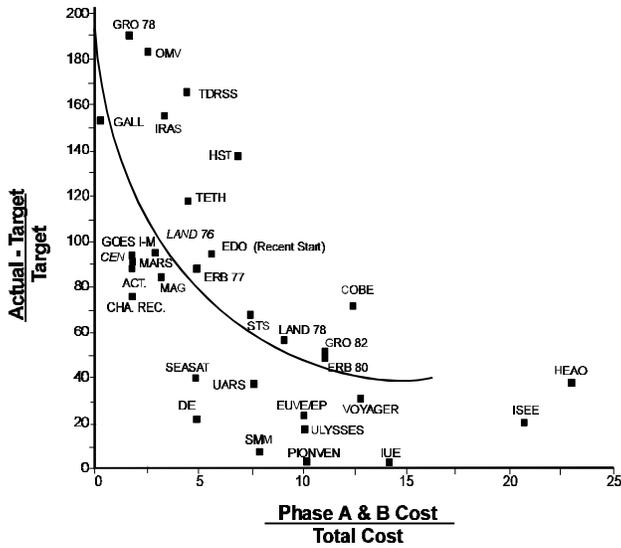
Figure 1. Effect of Requirements Definition
Investment on Program Costs
(By Werner M. Gruhl, Chief Cost & Economic
Analysis Branch, NASA Headquarters)

Therefore, I interpret the Gruhl chart differently. If you have not done a good job in Phase A and B in defining and confining your program, including documenting the requirements, you are going to encounter large numbers of changing requirements and the cost will go up accordingly.

The relationship between program cost and requirements is cyclic (Figure 2), you cannot affect one without affecting the other -- but program managers try. Budgets are cut, but the program manager tries to keep the requirements intact. There are some occasions where a design change will save money and all requirements will still be met, but this a rare occurrence.
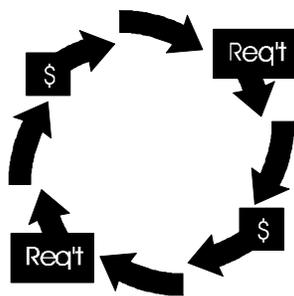

Figure 2. Cyclic Effect

It is almost impossible to change any requirement without affecting the net cost. Unfortunately, it seems that this is heavily biased in one direction, i.e., any change to a requirement results in an increase in cost. Even when you delete or reduce a requirement, you will encounter some cost -- you cannot make a change without paying. Hopefully, deleting or reducing a requirement will result in a net savings.

It seems obvious that requirements drive program costs and that changing requirements are a major driver of cost overruns. Poor requirements contribute to the need for change.

It is important to understand the type of errors that occur in requirements in order to avoid these errors and subsequent changes. An IEEE study (Figure 3) shows types of non-clerical requirement errors. In this study, ambiguities and inconsistencies make up about 20% of the errors and omissions account for another 31%. The largest number of errors 49% were for incorrect facts. Most of the incorrect "facts" that I have encountered come from incorrect assumptions.

The cost of assumption error chart (Figure 4) has been presented by many different companies and organizations over the years. The chart shows the relative cost over the software life-cycle to correct an "assumption error." If identifying and correcting the error during the requirements definition phase cost you $1.00, it will cost from 40 to 1000 times as much to fix if not identified until the operations phase. The cost to fix the error rises rapidly as you proceed into the life-cycle. I suspect that you only need to add a few zeroes to the multipliers to reflect the cost for hardware programs.
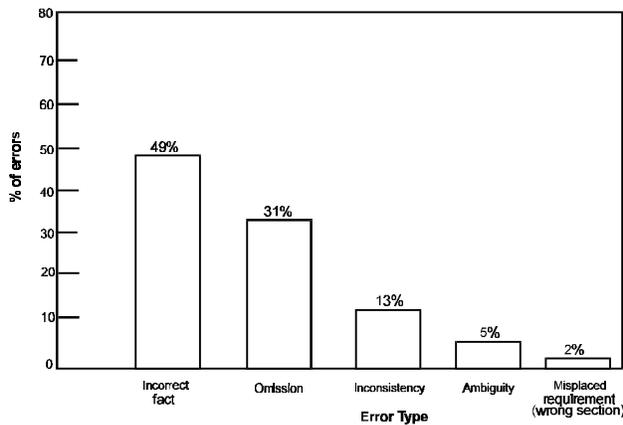
2

Figure 3. Types on Non-Clerical Requirements
Errors
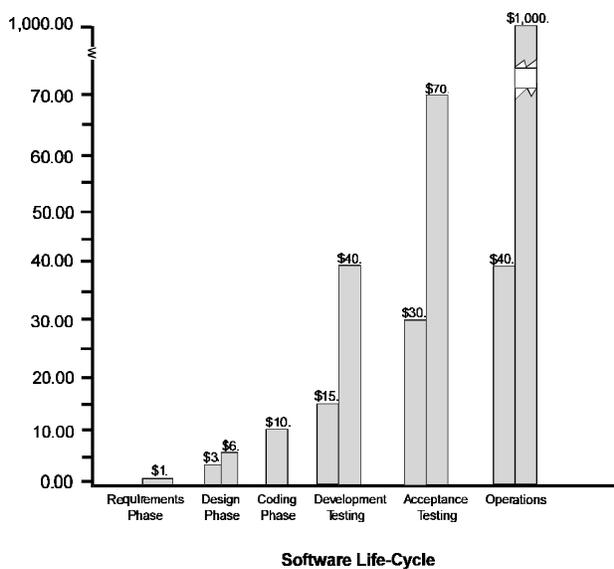(1981 IEEE Computer Society, Inc.)

Figure 4. Cost of Assumption Error in
Requirements Phase
("Extra Time Saves Money," Warren Kuffel
Computer Language, December 1990)

The information in this figure is also applicable to other requirements changes. If you decide to change a requirement at the beginning of the program, the cost will be minimal compared with making the change after you have begun development or when you are in operations.

These two previous figures indicate the importance of controlling all assumptions and all requirements from the beginning of the program. Gruhl's chart shows the importance of devoting resources to Phase A and B efforts.

Given the evidence of poor requirements definition and management as the cause of program cost overruns, why do program managers continue to make the same mistakes?

▪ Major Problems in Requirements Management

The major cause of bad requirements is that people do not know how to write requirements. The problem is compounded by a lack of management attention and a poorly defined requirements management process. If the program manager assumes that (1) everyone knows how to write requirements, (2) the requirement definition process is well understood, and (3) the review process will fix any problems, then problems are guaranteed.

1. **Everyone does not know how to write requirements.**
Not everyone, in fact very few people, really understands how to write good requirements. In each of my courses, I ask the class, "How many of you have had to write requirements?" then "How many of you have had to review or verify someone else's requirements?" Most of the class responds to one or both questions. Then I ask, "How many of you have been happy about either process?" Only on rare occasions does anyone respond to the final question.

The problem is that, while these are very bright people, they (1) sense a lack of management interest, (2) are not provided the information needed to do a good job, and (3) do not have the knowledge to do the job.

Lack of Interest. Requirement authors sense a lack of management interest. Emphasis is on schedule -- getting a specification written so that a procurement can be conducted -- not on quality. Most have never seen anyone recognized for doing a good requirements writing job and none have seen anyone suffer for having done a poor job. Hence, they just do the best they can, given limited information, time, and guidance. It is not surprising that the results are requirements that will need to be rewritten many times before the program is complete.

Nearly 1,500 NASA and contractor personnel have been through our Requirements Definition and Management Course. A recurring response to the post-class survey is "my management does not understand this process" and "my manager does not support my doing this work." This should be a red flag to all NASA managers.

Lack of Information. The NHB 1720.5 requires documentation of the scope of large programs and projects. The Program Plan is essential for all size programs and projects. Without this information, it is impossible to get good requirements. No one can write good requirements without a clear understanding of the scope of the project, its mission, and operational concepts. Each requirements author needs to know the goals, objectives and constraints associated with the program.

In fact, no one can write good requirements in a vacuum. If the program manager does not supply the scope, each individual author will define a scope. Each individual will probably define a unique scope -- the resulting requirements will be responsive to a variety of concepts, objectives and constraints. This is an invitation to disaster.

NASA has established the process, but it is up to each individual program and project manager to ensure that the content, quality, and timeliness of the Program Plan supports the requirements development process.

Lack of knowledge. Engineers at NASA frequently are asked to write, review, design to, or verify requirements very early in their careers. They may not have ever heard the word "requirement" in college. They are sharp so they have an idea of what they are to do, but their ideas and examples of existing requirements may be all wrong. If management is not prepared to mentor and assist these new engineers, they will do their best, but it will not be good enough.

Some people with many years of experience still do not have an appreciation for either the importance of good requirements or what it takes to write good requirements. Some of these people may be trying to mentor, but they also lack the necessary knowledge.

Recently, a division chief was reviewing a report that I had written against a set of system requirements. The report showed the current requirement, explained what was wrong with it, and provided a rewrite. His response was, " I would have thought these current requirements were okay." He was just being honest, although he lacks the knowledge to help his people. In fact, the lack of sufficient and knowledgeable mentors has affected all levels of NASA personnel.

Only requirements that are necessary, attainable and verifiable should appear in a specification. If the requirement authors are appraised of this and held accountable, there is some chance of creating a valid specification. Each of these attributes is essential to good requirements, and further details are provided later in this paper.

2. The requirement definition process is not well understood.

Many view the requirement definition process as only major milestones -- release of a specification for the Request for Proposal (RFP) and a System Requirements Review (SRR). The process involves many steps to reach the milestones, but these are often ill-defined or not communicated to the team.

A disciplined program manager must assure that the steps are clearly defined and communicated, ample time is allotted, and a qualified team assembled to ensure a good specification. Otherwise, the result will be a poor specification, tons of Review Item Disposition (RID) forms, and more effort in the review process than was ever expended in the requirement definition process.

Too many cooks can spoil the broth, especially if each is working to a different recipe, i.e., working without a well defined Program Plan. The NASA approach to requirements is too often to invite everyone to create a wish list. You will get unnecessary, unverifiable and unattainable requirements if you do this. If you want to solicit requirements from a large group of people, you must provide them the Program Plan and insist that their requirements be responsive to your plan. You must instruct them to justify each requirement, just as you will require them to justify each future change. You must educate them about defining only requirements that are necessary, verifiable and attainable.

You need to use concurrent engineering in defining requirements. This is essential to ensure that all requirements are captured in the initial definition phase, not after design, testing, or operations are underway. This means having not only the customer, user, and functional area designers involved in the process but also participants from safety, reliability, manufacturing, test and operations.

Failure to include this cross-discipline group of people in the requirements definition process can result in a system that exceeds costs for manufacturing, is unreliable, and that will cost a fortune to maintain and operate. Too many problems will be found too late in the program life-cycle, and the program costs and schedules will overrun significantly, as indicated in Figure 4.

The requirement definition process needs strong, experienced, system-oriented personnel to help elevate detailed engineering discipline requirements to real system requirements. Discipline engineers will tend to write requirements as though for their discipline, resulting in detailed subsystem definition before the system design is done. It is not unusual to see a system specification with requirements that read:

The guidance and navigation subsystem shall......
The failure and warning system shall ....
The communications subsystem shall..

These are not system requirements and they play havoc when a contractor designs your system and develops lower level requirements. A strong system engineer can assess the real needs and develop system level requirements from those proposed by discipline engineers.

Requirements defined by scientists also require a good system engineering to interpret and translate science requirements into engineering requirements. Many of the NASA Centers handle science requirements, and the subject arises repeatedly in our training classes. The engineers are frustrated in two areas. They see no constraints on the science requirements -- they can be simply a wish list. In fact some scientists seem to feel that they are entitled to ask for anything on a NASA program, since they do not have to pay for it. Management must control the science requirements just as rigorously as engineering requirements -- Are they necessary? Are they attainable?

The second frustration is one of translating science requirements into engineering requirements. Centers that repeatedly face this challenge need a team of experts to do this job. Scientists know what they want but are often unable to write an appropriate statement for a specification. Engineers who understand what the scientist wants and can translate this into a

valid and verifiable system requirement are very valuable.

To ensure proper translation, each requirement written in response to a science requirement should clearly document the assumptions made and how the translation was conducted. Then the scientist should be asked to approve the engineering requirement(s) and review the operational concept and implementation before baselining.

It is important to select the right team of people and to put in place the right processes with reasonable schedules in order to succeed in the requirements definition phase of the program.

3. The review process cannot fix all problems.

If you have (1) produced a very good set of requirements, (2) selected knowledgeable people for the review process, and 3) managed the review properly, you will be rewarded with a set of recommendations to improve your program. If you have failed to do any one of these steps, the review process will largely be a waste of time and money.

The review completion allows you to move into the next phase of the life-cycle. But a review of poor documents, no matter how well-conducted, will increase your program risk. You will not have identified all the necessary items and you will be redefining throughout the next phase, leading to increased costs and schedules slips.

Some large NASA programs have recently had upwards of 7000 RIDs against a single document. This is inexcusable. A number of factors contributed to the problem. The first and foremost problem was that the document being reviewed was in too poor of shape to have been released in the first place. It should have been cleaned up considerably before being allowed out for review. This is clearly a management problem.

Secondly, there were too many and too inexperienced reviewers. Managers have told me that they had no control over who reviewed the document. This is ridiculous. This is a program cost and it should be controlled. Many of the reviewers have stated that it was expected of them to write a certain number of RIDs -- this is unbelievable. The reviewers were often inexperienced and so wrote individual RIDs for every editorial comment -- this will certainly get the numbers up.

Management should provide instructions for the review. This should include stating that all editorial RIDs can be placed on a single form. You might question why, with grammar and spelling checks available on all word processors, there are any editorial RIDs at all. All participants in the process should be qualified as having some knowledge in both the process and the program before they are allowed to write RIDs. This make take some effort on the part of management, but not nearly as much effort as struggling through hundreds of useless RIDs.

Improving the Requirement Management Process

Steps to improving requirement quality and the requirement management process are straight-forward and can be implemented with minimal cost and extraordinary results.

The first step is to ensure that a good Program Plan, containing goals, objectives, constraints, missions and operational concepts, is available to all participants. The second step is to establish a well defined requirement definition process and to educate the participants to their responsibilities.

It is essential that each participant be aware of the characteristics of good requirements -- necessary, verifiable, and attainable. The requirement definition process must include tests of these characteristics..

Necessary. I once requested that an engineer withdraw a requirement, since it was unnecessary. The engineer said, "No, let my manager take it out if he wants to." Odds are the manager will not catch the problem. Responsibility, authority, and accountability must be identified and enforced.

Responsibility should be imposed at all levels, but it ultimately comes back to the program manager. The necessity of every requirement should be clearly understood before the first draft is released -- not during CDR.

Each requirement proposed for a specification should be examined as rigorously as each change will be examined in the future. The first time a requirement appears, you should treat it as though it were a change that will cost your program a great deal of money. You need to know why the requirement is needed and any assumptions that were made by the author. These are questions you will ask for each change -- ask them now.

All requirements should be in response to your Program Plan. If they are not, they may not be necessary.

Attainable. It is a waste of time and money to write unattainable requirements. If the effort is for new technology, then there may be a question about the technical ability to attain the requirement. This can be handled by tracking the requirement as a risk.

Unattainable requirements often come into being because the original author does not know what is needed. The Space Station requirements have been through many iterations. Unfortunately, no rationale or justification was captured in the process. As some items are converted from contract to GFE, it has become apparent that unattainable requirements were written and never caught. One recent problem requirement affected the use of the Global Positioning Satellite (GPS). The requirement was for an accuracy unattainable by the GPS. When questioned, it was divulged that no one had computed a required value, an engineer had simply guessed that a certain value was attainable and entered it into the requirements. Management had not questioned the value. The requirement will have to change and someone will need to determine the correct value. Remember Figure 3 -- 49% of the requirements errors were incorrect "facts"?

Many unattainable requirements are technically feasible but still unattainable. You do not need requirements that exceed your budget -- even if they are technically feasible, they are unattainable. Unmanaged authors will write requirements for many items that would be "nice to have" but are really unnecessary or unattainable due to budget and schedule constraints. It is the job of program management to prevent this occurring.

Verifiable. It is hard to believe that there are engineers and managers who do not know that all requirements must be verified. It is important to analyze each requirement in light of how it will be verified as it is written and before it is baselined. This is not the case on all programs. Last year a change request was written for the Space Station Freedom Program to correct or delete over 100 unverifiable requirements from the system specification. One can only wonder how more than 100 unverifiable requirements had remained in the document through so many reviews and scrubs.

A good check against unverifiable requirements is a simple test of word usage. Words and phrases like maximize, minimize, support, adequate, but not limited to, user friendly, easy, and sufficient are subjective and thus unverifiable.

Verification costs are often a major element of the program cost. Removing unverifiable requirements and specifically addressing how

each requirement will be verified, prior to baseline, can help to control this cost.

The most significant step that needs to be taken in improving the requirement process is that of accountability.

Accountability is important for each individual requirement. You need to assign ownership as requirements are written. The owner should be a person with a stake in the requirement and who is knowledgeable about the need for the requirement. The owner should be willing and able to defend the need for the requirement prior to baseline. The owner should be available to assess change impact against the requirement should a change be proposed.

Accountability is even more important at the management level. There has been a trend for large numbers of people to sign a specification. I have seen instances where a division chief, an associate director, and the director signed the specification -- but not the program/project manager. These signers had not read the document. The program manager should sign and be held accountable. Higher managers can sign if they wish, but if they sign they should be held accountable.

The quality of the requirements should be part of each program manager's evaluation criteria. The quality and stability of the requirements that they manage are essential to program success and should be a measure of the their own success.

Anyone offered a program manager's job should look carefully at the condition of the requirements left by the predecessor. If the requirements are out of control, no other control, short of cleaning up the requirements, will enable the program to be successful.

What all program managers should recognize is that the investment to obtain good requirements is minor compared to the effect on program cost and schedule, and possibly the manager's career.