Managing Software Requirements: Organizational and Political Challenges

by

Laurie L. Rodriguez

MSc Analysis, Design, and Management of Information Systems, The London School of Economics, 2001
B.S.E. Electrical Engineering, Princeton University, 1999

Submitted to the Engineering Systems Division and the Department of Aeronautics and Astronautics
In Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Aeronautics and Astronautics
and

Master of Science in Technology and Policy

at the

Massachusetts Institute of Technology

February 2004

© 2004 Massachusetts Institute of Technology All rights reserved

Signature of Author

Laurie L. Rodriguez
Department of Aeronautics and Astronautics
January 26, 2004

Certified by

Charles P. Coleman
Thesis Supervisor
Boeing Assistant Professor of Aeronautics and Astronautics

Accepted by

Dava J. Newman
Director of the Technology and Policy Program
Associate Professor of Aeronautics & Astronautics and Engineering Systems

0
Edward M. Greitzer
Chair, Committee on Graduate Students
H.N. Slater Professor of Aeronautics and Astronautics
Abstract

Requirements management is an extremely critical and challenging part of the software development process. Correctly capturing these requirements is necessary for the user to be satisfied with the final product. Many instances of inadequate requirements management result in subsequent problems with the final product and have organizational sources. The goal of this research is to better understand the organizational processes and problems associated with software requirements management within NASA’s human-rated space programs through interviews with experienced professionals within this organization.

Some of the major themes that resulted from the expert interviews include:

- Software engineering practices, such as the CMM, are starting to be implemented for the larger programs at NASA, however these practices have not been equally pervasive in all parts of the organization.
- The main reason for lack of requirements management at NASA is not enough time or people available for the activity.
- The reason that was most cited for requirements management tools not being used effectively is that these tools are too manpower intensive for NASA’s current organizational situation, and require documentation that is not available.
- To achieve improvements in requirements management NASA should hire more people with computer science backgrounds who also have an understanding of aerospace systems.
- The lack of complete documentation on NASA projects, means that complete system testing cannot happen.
- Requirements creep can happen at NASA because higher level customers do not always have a complete technical understanding of a subsystem or component that is
being developed, and the designers may interpret or change a requirement without consulting the customer.

- NASA often deals with managing relationships between different subcontractors who have responsibility for different parts of the software lifecycle, and each subcontractor has its' own interests and stake in the final outcome of the system. It is important that NASA adequately manages the requirements between these parties to ensure that the customer’s system requirements do not become distorted by the political interests of the subcontractors.

Finally some recommendations for further research in this area are made.

Thesis Supervisor: Charles P. Coleman
Title: Boeing Assistant Professor of Aeronautics and Astronautics
1. Introduction

“If we want to improve practice in some way we must, first of all, acquire a more comprehensive understanding of the work processes of real-life design and programming, including not only technical aspects, but also behavioral and sociological aspects” [1]

Requirements form the foundation for a software system. Ideally, good requirements are an agreement among developers, customers, and users on what the system must do and on the acceptance criteria for the delivered system. Good requirements can provide a sound basis for resource estimation and allow project managers to plan ahead and achieve goals with a minimum cost, personnel quantity and skills, equipment, and time. From requirements, come test plans and test cases to determine that the software does what the user wants. Without a solid requirements backbone, there will be problems later on in development, as resources will be wasted, and testing plans will not properly validate the user’s conception of the software system.

The benefit of having good requirements is clear. In fact, more than 20 years ago, requirements analysis was identified as a key issue in the success of software development and maintenance. [2] Since then a lot of software engineering literature, and studies have been aimed at discovering the barriers to good requirements management, and how to address this with tools, organizational practices, and process methodologies. However, many organizations still suffer from poor requirements management, and requirements creep.

Requirements creep is a tendency for product or project requirements to increase during development beyond those originally foreseen, leading to features that weren't originally planned and resulting in risk to the product’s quality or schedule. It is a problem that often leads to project lifecycles running over their planned schedule and budget, and
delivered products that do not satisfy the user. Because requirements creep, and requirements management are such a challenge for organizations, research into how organizations manage requirements can lead to valuable knowledge for the organization. The goal of this thesis is to examine the state of software requirements management within NASA’s human-rated spacecraft programs, with a focus on the organizational processes that determine how well requirements management is occurring. The research was carried out through phone interviews with people that have expertise at managing software requirements within NASA. Questions were designed to determine the nature of the requirements management that is taking place within this organization, and identify obstacles to improving requirements management within NASA.

The organization of this paper is as follows. The next chapter outlines the motivation for this research in further detail, making it clear to the reader how influential organizational processes are on the final success of requirements management. It is shown how requirements creep can originate from many different places in an organization when a software project involves many stakeholders, and, thus, why it is necessary to look for organizational solutions to requirements creep. Lifecycle models such as the spiral model, and software engineering practices and tools are discussed to give the reader an idea of what is possible for requirements management. Chapter 3 discusses the research method that was used to do a case study of software requirements within NASA’s human rated spacecraft programs, and why this method was chosen. Chapter 4 and 5 present the results of the research. Chapter 6 summarizes the findings and presents further directions for this research. It will also explain what these results mean for software requirements management within NASA, and relate the results to current literature in software engineering.
2. Motivation

Establishing requirements for a software project is an extremely critical and challenging part of the development process. Correctly capturing these requirements is necessary for the user to be satisfied with the final product. With the goal of creating a product that will satisfy their needs and succeed in their organization, users often change requirements, sometimes even late into the development cycle, causing requirements creep. Software developers, and customers alike, often complain that conflicting requirements can cause a project to go over schedule and cost, thus increasing the chance of project failure. For the project to succeed, a balance must be reached between the extent to which users can change requirements, yet still allow the developers to complete the project within the initial cost and schedule. This is the goal of requirements management.

In general, a software requirement should be cost effective, meet the end users needs, be complete, and be traceable. Traceability refers to the ease of locating the source of the requirement and where and how it is implemented in the system. Good requirements can provide a sound basis for resource estimation (cost, personnel quantity and skills, equipment, and time) and is the only way to truly allow project managers to plan ahead and achieve goals with minimum resources resulting in less rework, fewer omissions and less misunderstandings.

The benefit of establishing good requirements has already been stressed: software systems are very sensitive to the quality of their requirements and many errors in operational software can be traced to errors in requirements. However, most software engineering methods focus on software design and coding, yet few techniques exist for validating requirements. In addition to this little contemporary data exists to document actual practices
of software professionals for software requirements elicitation, requirements specification
document development, and specification validation. [3] The many challenges to good
requirements management, including the sources of requirements creep will now be
explained.

2.1 Fluctuating and Conflicting Requirements

Requirements change can cause havoc in a development process when the effects of
the changes ripple through a large part of the system and software design and affect other
parts until the overall design and conceptual model may be affected and start to degrade.
Results of a case study [4] listed fluctuating and conflicting requirements, communication and
coordination breakdown, and the thin spread of application domain knowledge, as the three
most important problems in software development.

There can be many sources for requirements fluctuation such as market factors and
technological advances. In addition, internal company factors such as corporate politics, and
financial conditions can also cause requirements creep. Product requirements often fluctuate
when different customers have separate needs or when the needs of a single customer changes
over time. And this type of project is more likely to run over schedule or cost, as “Many
customers misunderstood the tradeoffs between requested functions, the capabilities of
existing technology, the delivery schedule, and the cost,” and customers rarely understood the
impacts that rippled through the software when changes were made. [5] Instead, customers
often learned of these tradeoffs during the project development, through an iterative
negotiation with the system design team, as the requirements were translated into a design and
costs that could be estimated. Therefore the customer wanted to change a requirement without realizing how much extra time it will take to manage and implement the change.

The following chart was created from data from the previously cited case study and depicts potential organizational sources for conflicting requirements:

<table>
<thead>
<tr>
<th>Individual Level</th>
<th>Some requirements are unclear and subject to interpretation. Individuals add system features that are not part of requirements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Level</td>
<td>Design team interprets conflicting or unclear requirements responsible for different components do not coordinate the assumptions they make about requirements.</td>
</tr>
<tr>
<td>Project Level</td>
<td>Absence of defined mission for project. Requirements fluctuated with prevailing attitudes of those who approved funds.</td>
</tr>
<tr>
<td>Company Level</td>
<td>The requirements and understanding of the product can vary among different groups in the company (marketing, vs. designers).</td>
</tr>
<tr>
<td>Business Environment</td>
<td>Different customers have different needs or needs of single customer changes over time. Customers learned more about the system’s capability and technical and non-technical constraints</td>
</tr>
</tbody>
</table>

Now that it has been shown why requirements management is a challenging process, and how requirements creep can originate from many different levels and sources within an organization, this paper will turn to discussing how some common models and frameworks for software development assist in managing software requirements. The main goal of this chapter was to give the reader the understanding that organizational issues are key in managing requirements, and is the reason that this is the focus of this research.
2.2 Models for Improving Requirements

This section will discuss some software engineering models and frameworks for software development that can be used to improve requirements management: the spiral model for software development, and the Capability Maturity Model (CMM). After these models are presented, requirements management is explained within the context of these models.

The figure on the following page shows the spiral model for software development. [6] There are other models for life-cycle development such as the waterfall model, and the evolutionary model, but this model allows an organization to tailor the lifecycle to suit individual circumstances.
The goal of this model is to manage the software development process in a way that allows for mitigation of risk. This model places emphasis on the use of prototypes in an incremental fashion to flesh out uncertainty. Each cycle begins with the identification of alternative means to implement this portion of the product and identifies sources of project risk. Then a prototype is made and validated and the next cycle begins. The process is flexible to incorporate lifecycles where there is a lot of uncertainty in what the user wants, and
this uncertainty is resolved as more prototypes are built. On the other hand, a project that is well defined can skip these steps and follow the more traditional waterfall model, which is depicted in the outermost spiral of this diagram. Lifecycle cost is measured on the x-axis, and increases with each spiral. This model can work well for requirements management because an evolutionary approach can be a good way to manage requirements and build stakeholder confidence in the product as the development proceeds.

Another software engineering practice that is used to manage the software development process is the Capability Maturity Model (CMM). The CMM states that the ideal software process should be predictable - cost estimates and schedules will be met with consistency and the final product will meet user needs. This model uses a framework of five levels consisting of initial, repeatable, defined, managed, and optimized. Using the CMM’s assessment, an organization can determine it’s current level, and focus on key process areas and goals to increase in process maturity. Highest levels manage software development by a process database with information on costs and benefits of major process activities.

Many organizational issues have been incorporated into the CMM. For example Requirements Management and Sub-contract Management are listed as key process areas under level 2. Focusing on these processes will lead to establishment of a common understanding between customer, sub-contractors, and developers, and it is essential to do this if an organization wants to proceed past the initial ad-hoc level of software development.

Correct application of these models can improve management of the software development process, and therefore of requirements management. Computer Aided Software Engineering (CASE) tools have also been introduced within these lifecycle models to help organizations achieve the goals of these models and frameworks. There are several
Requirements Management tools that are on the market, one of which is discussed next in this chapter.

2.4 Requirements Management Tools

DOORS is a requirements management tool that was sited in some of the expert interviews. [7] DOORS is a tool that is designed to capture, link, trace, analyze and manage a wide range of information to ensure a project's compliance to specified requirements and standards. DOORS can facilitate communication and coordination within an organization through a user interface that makes access easy for large numbers of concurrent users on a network. In addition DOORS can maintain vast numbers of objects including requirements and associated information and attributes and links to improve traceability. DOORS also includes a complete on-line Change Proposal and Review System that lets users submit proposed changes to requirements, including a justification. Inter-project linking that allows projects to share requirements, designs and tests, and promotes traceability to corporate or other standards. Discussion threads allow subject-oriented collaboration on ideas. And Distributed Data Management supports remote users who need temporary, remote access to all DOORS features. Working against a subset of the DOORS database off-line, remote users can incorporate their updates back into the master database making it easy to team with other organizations to communicate with subcontractors and suppliers. DOORS also provides a way to improve the link between requirements and verification of these requirements, which is important in verification of the final product. DOORS also provides user-defined, multi-level traceability for unlimited relationships. For example, requirements to test, requirements to design, design to code links are provided.
The purpose of this explanation was to expose the reader to some of the features that requirements management tools can provide. Requirements management tools can be introduced into a mature software development process to address many of the organizational challenges to requirements management. Whether they are right for one’s organization or specific project will be a decision based on specific organizational and project characteristics such as size, maturity of software engineering practices, and the distributed nature of the parties involved in the process.

This chapter has presented the motivation behind this research by showing how requirements creep can be a problem in the software development lifecycle, and how organizational issues can produce many challenges to requirements management. Software engineering practices such as the CMM and spiral model, and requirements management tools can help organizations effectively manage requirements, however they are not effectively used by all organizations, therefore they do not tell the whole story. In order to understand to how well an organization is employing these methods and tools, and how well they are managing requirements creep, it is necessary to look within the organization, and learn about their current practices. That is what we will attempt to do with NASA’s human-rated space programs.

3. Research Methodology

3.1 The Value of Understanding Organizational issues in Software Development

Previous studies have demonstrated the substantial impact of behavioral (i.e. human and organizational factors) factors over technical tools, on software productivity. In fact, the effects of tools and methods were relatively small in these studies. Studies have found that applying a collection of software engineering technologies to actual projects had only a 30
percent impact on reliability and none on productivity. [8] Bill Curtis states, “To create software development technology that dramatically improves project outcomes we must understand how human and organizational factors affect the execution of software development tasks.”[9] Therefore, in order to really understand how requirements decisions are made, communicated, and changed, it is important to conduct field tests in ongoing software development projects. The goal is that by conducting interviews and distributing a questionnaire with people in various roles on a software project, the behavioral and organizational processes that control requirements generation and their connection to project success will be better understood. Eventually these best practices can be incorporated into models that are used widely in industry such as the CMM. This research approach was used within NASA, to learn more about how they manage requirements.

3.2 The Research Method

In order to capture expert requirements management knowledge within NASA, five phone interviews were conducted. The roles of the people that were interviewed within NASA were mostly in the project/program software management level. The sample was not large enough to produce statistical data, but there can still be useful qualitative information drawn from the information from these interviews. The questions asked during these interviews were inspired by a questionnaire that was initially developed for wide distribution within NASA. The questionnaire was designed for a larger sample in to focus on gaining a broad, exploratory understanding of the organizational and political issues that affect requirements management, with the goal of getting some statistical data. Due to time constraints this was not explored, however could be the starting point for doctoral research.
These phone interviews serve as a starting point for the data collection, and the questionnaire is presented below to give the reader an idea of the types of questions and discussion topics that were covered in the interviews.

3.3. Questionnaire

**Section 1: Background Information:**

Full Name (optional):
Date:
Organization:
Job Title:
Number of Years Working in Current Job:
Area of Expertise:

**Section II: Organizational Factors:**

1. Who is involved in the requirements process (Project Sponsors, Project Managers, Developers, Users, Customers)?
2. Ideally, who do you think should be included in the requirements process? What are the barriers to getting all the right people involved in the process?
3. Do your requirements come from more than one source? How do you deal with inconsistencies?
4. What organizational process do you use to define the initial requirements (workshops, meetings)? What works best for your process and what are the main problems? Some things to consider are:
   a. Do you go into a requirements planning meeting with an agreed-upon deliverable? Do the participants have roles and assignments?
   b. How long does it typically take you to define the initial set of requirements? How many meetings occur and how often do they occur?
   c. Do you use requirements analysts, or designated people to play a bridging role between technical developers and customers and users?
5. How well are the people that have a stake in the project requirements kept informed of the requirements? How do you make up-to-date information on requirements available throughout the project?
6. What type of communication (formal meetings, informal meetings, e-mail etc.) do you think works best for establishing requirements?
7. What types of analytical tools do you use to document and model requirements, and how effective are these tools in managing requirements?
Section III: Political Factors:

8. Are there problems identifying the stakeholders in a project? Are there problems identifying who has ownership of the project? Does it change according to the political favorability of the project?
9. Are there stakeholders who have a vested interest in seeing the project fail or succeed? How has this affected project success in the past?
10. How do you form support for the requirements of a project across your organization?
11. What are your best practices for making requirements decisions within your political environment? How well are these lessons transferred to future projects?

The next chapter will qualitatively present the results of the interviews, and following this some themes and findings from the interviews are discussed.

4. Software Requirements Management within NASA’s Human-rated Space Programs

4.1 NASA Software Policies

At the Fifth International Symposium on Requirements Engineering Dr. Linda H. Rosenberg stated that in many past development efforts at NASA, requirements were paid little heed. Dr. Rosenberg is the Chief Scientist for Software Assurance at NASA’s Goddard Space Flight Center. She states that “at NASA, in recent years, the hue and cry for project development has been “Faster, Better, Cheaper and Safer”. [10] This has impacted the way we develop software; it has increased the risks to quality, safety and reliability.” Recently NASA has tried to change this, and the Software Assurance Technology Center (SATC) is working with projects to emphasize the criticality of requirements throughout development, not just in the initial phases. The emphasis is on the requirements relationship to all aspects of quality, including reliability and safety. This is part of the NASA Software Engineering Initiative.
In 2002, the NASA Software Engineering Initiative had been in place at NASA for one year. This initiative is a coordinated effort to improve the software engineering capability, and is sponsored by NASA’s Office of the Chief Engineer. All 10 NASA Centers participate in this initiative. Software engineering is now recognized as a core capability and a key enabling technology necessary for NASA’s success. The NASA Software Initiative relies on the Software Working Group as an advisory group who implements Agency & Center level software engineering improvement plans. Now assessments using CMM/CMMI identify areas of strength, and allow infusion of best practices of software engineering research and technology.

The following policies cover software created and acquired by or for NASA and also cover Government off-the-shelf (GOTS) software and commercial off-the-shelf (COTS) software when included in a NASA system.[11] It is stated that these policies shall be applied as appropriate, consistent with sound engineering and risk management practices as determined by cost, size, complexity, life span, risk, and consequences of failure. As stated in this document, NASA policy regarding software management, engineering, and assurance is to accomplish the following:

1) Manage, engineer, and assure software in accordance with common industry standards, processes, and best practices; document the use of standards, processes, and best practices in accordance with ISO 9000; and tailor standards, processes, and best practices to the development or acquisition.

2) Implement and integrate software engineering processes and practices with other system development and program/project processes and practices.
3) Develop a plan for acquisition and life-cycle management of the software as part of the program/project plan. This plan will be developed prior to selection of the provider and will address design tradeoff management, risk management, requirements management, software project planning, project tracking and oversight, software product engineering, subcontract management, configuration management, quality assurance, and peer review.

4) Develop and maintain a total estimated software life-cycle cost and perform tradeoff studies which address use of COTS and GOTS software versus created software to satisfy requirements before software is created or acquired.

5) Demonstrate that the provider of software to be developed has proven organizational capabilities and experience to deliver quality software on time and within budget. Examples of current acceptable evidence include an independent certification of ISO 9001 compliance as described in ISO 9000-3 or an independent assessment of a software Capability Maturity Model (CMM) rating of 3 or above. The provider shall develop a plan to manage software throughout the program/project life cycle before the software requirements specification is complete and software design and coding takes place.

6) Document software as to its form and function and verify that such software performs the functions claimed on the platform(s) for which it is designed without harm to the systems or the data contained therein.

7) Develop risk analyses and management strategies; identify, analyze, plan, track, control, and communicate risks at each stage of the life cycle. Document or reference (i.e., their location specified) the results of risk analyses and management strategies in program/project plans; and employ verification and validation techniques for risk mitigation,
including Independent Verification and Validation (IV&V), as appropriate, based on cost, size, complexity, life span, risk, and consequences of failure.

8) Facilitate reuse of NASA-funded software, as well as transfer, consistent with law and applicable agreements, for commercial, industrial, educational, and governmental purposes; and protect NASA-funded or -created software as valuable intellectual property during all phases of the lifecycle.

This policy document shows that NASA is aware of the importance of using software engineering processes and practices to have good life-cycle management, and requirements management for software development. The use of lifecycle cost estimates and trade studies to determine whether to use COTS or to develop the software, is being performed at NASA. They also recognize the impact that organizational maturity has on the ability of the developer of the software to deliver a product on time and within budget. This is shown through the statement that the developer of the software needs to document, and have a way to plan, track and control each stage of the lifecycle. Mature requirements management must be present within the contractor’s organization.

While this is the current policy at NASA, just how well is NASA using the techniques from software engineering? To what extent is NASA holding the contractors accountable to these requirements management practices, practices that were described in Chapter 2 of this paper? How well are these techniques and processes being used throughout NASA programs to manage requirements, and what are the organizational barriers to these requirements management processes becoming more pervasive? In other words, what is the reality of current state of requirements management within NASA’s human-rated space programs?
These are the questions that this paper will attempt to begin to address. Some of the main findings from the interviews are highlighted in bold.

4.2 Results from Expert Interviews

One finding from the expert interviews is that while software engineering practices are definitely starting to be implemented for the larger programs at NASA, smaller projects, that may even support the larger projects, have not received the same attention. Expert interviewers say that even for large projects like the Space Shuttle and the International Space Station (ISS), requirements creep did happen. The projects suffered from this, as requirements never stopped being added to a project. When NASA handed these requirements off to a software workshop they did not make sense. Requirements kept growing at a continual cost. However now that the new Software Engineering Initiative is being put in place, requirements creep is becoming more under control. In fact, now the large human-rated space projects such as Shuttle and Station have large organizations to deal with requirements management, and the CMM is being used within these organizations. For example, man-rated flight software requirements are being written down, evaluated and costed.

However, the CMM and Software Engineering Initiative are not pervasive throughout the whole organization. In particular, good requirements management is not common practice for the smaller projects and programs. Smaller programs sometimes do not even manage their requirements at all. The main reason that was given by the expert interviewers is not enough time or people. An example is a small project to design a simulator which will be used for modeling purposes within NASA. On this type of project design teams are going
as fast as possible to get the models working, however not enough attention is paid to requirements management, or documentation. How much does this matter on a small project like this? One expert interviewer said that even on small projects NASA pays for a lack of requirements management when things don’t work, and the architecture does not support future change.

When it comes to the use of requirements management, tools such as DOORS and Rational Rose were mentioned as being used on some of the larger projects. However these tools were not used as effectively as they could have been. **The reason that was most cited for requirements management tools not being used effectively is that these tools are too manpower intensive for NASA’s current organizational situation, and require documentation that is not available.** Rational Rose will take a word document and parse it. However this requires a lot of manpower to maintain documentation, and this does not usually happen: “With hardware there will always be drawings, but with software there is not always the manpower to maintain the drawings.” [12] In fact, Shuttle and Station may have documented the requirements but do this well, as they did not document the rationale for their requirements.

So how did these programs manage requirements? The shuttle program managed requirements by building documents by hand, and tracing all relationships by hand. If the requirements are stable you can build models from this, however if they are not this method of documentation is not adequate. The ISS project used a more advanced method that had a database that would manage relationships between requirements, and capture new requirements and update the model. Overall, the interviewers did not think NASA took advantage of requirements management tools effectively, however it still remains to be seen
what the full effects of the Software Engineering Initiative will be. It may be the case that NASA will use these tools better in the future.

The interviews also pointed out that there is also a cultural problem within NASA that has prevented effective requirements management from happening. NASA hires mostly aeronautical engineers who are not good at chasing down requirements. There is a need within the organization to hire more people with computer science backgrounds who also have an understanding of software engineering and aerospace systems. This way the staff will be more interested and capable to work with the software engineering practices and document the requirements.

Another issue that was brought up by some of the expert interviewees is the link between requirements and testing. The lack of complete documentation on NASA projects, means that complete system testing cannot happen. If a requirement is undocumented it is untested.

The following are some commonly sited sources of requirements creep at NASA. The customer may work at a higher level of the organization, and have an idea of what he wants, but does not fully understand the technological reality of the specific system. Customers have to be smart customers. At NASA the requirements for a program such as the Orbital Space Plane (OSP) may come from Headquarters. They may not have a full conceptual technical knowledge of the details of the system, and someone at say, Marshall Space Flight Center, may have to interpret what HQ wants. When interpretations happen at a low in the organization, they may not correctly capture what the customer intended. The customer must be continually involved in this process so that they can tune their requirements to the technical reality. However, at NASA, the designer coming up with a requirement is a pretty common
feature. Therefore, requirements creep can happen at NASA because higher level customers do not always have a complete technical understanding of a system or subsystem, and the designers may interpret or change a requirement without consulting the customer.

Requirements management also becomes more difficult when there are multiple organizations involved in the process, and this paragraph involves a situation where this happened. This is a situation where different subcontractors were responsible for different parts of the lifecycle. During the development of the space station Freedom, the predecessor to ISS, one organization was doing the requirements and another the design for the software in the propulsion subsystem. The organization responsible for the requirements tried force a specific software design on the project through the requirements. This would have contributed to problems later in the lifecycle as one would be verifying the design rather than the requirements. NASA often deals with managing relationships between different subcontractors who have responsibility for different parts of the software lifecycle, and each subcontractor has it’s own interests and stake in the final outcome of the system. In these cases, it is necessary for NASA to manage the requirements between these distributed parties to ensure that the customer’s system requirements do not become distorted through the political interests of the subcontractors.

5. Research Findings

The following are the findings of this study of software requirements management within NASA’s human-rated space programs:
• Software engineering practices, such as the CMM, are starting to be implemented for the larger programs at NASA, however these practices have not been equally pervasive in all parts of the organization. The smaller projects do not manage their requirements.

• The main reason for lack of requirements management at NASA is not enough time or people available for the activity.

• The reason that was most cited for requirements management tools not being used effectively is that these tools are too manpower intensive for NASA’s current organizational situation, and require documentation that is not available.

• To achieve improvements in requirements management NASA should hire more people with computer science backgrounds who also have an understanding of aerospace systems and software engineering.

• The lack of complete documentation on NASA projects, means that complete system testing cannot happen.

• Requirements creep can happen at NASA because higher level customers do not always have a complete technical understanding of a system or subsystem that is being developed, and the designers may interpret or change a requirement without consulting the customer.

• NASA often deals with managing relationships between different subcontractors who have responsibility for different parts of the software lifecycle, and each subcontractor has its own interests and stake in the final outcome of the system. In these cases, it is necessary for NASA to manage the requirements between parties to ensure that the
customer’s system requirements do not become distorted by the political interests of the subcontractors.

Even though this is only the first stage that this research should take, findings were still produced from this approach through listening to what interviewees had to say about the most important problems in requirements management. The following chapter will do a brief review of some current literature to put these findings into context.

6. Summary

Sol Greenspan wrote a paper called “What if there is no time for Requirements Engineering?” [13] This paper discusses the need to find ways to do requirements engineering in the presence of severe time constraints, a condition that is present at NASA, as there is often a shortage of time and people to do requirements management. This article points out that organizations under tight time constraints are not finding the arguments for controlled processes and requirements engineering very persuasive. They do not see it worth their while to take the time to improve requirements engineering methods tools and practices, and when time is tight, requirements engineering activities seem to be among the first to be neglected. “If we listen closely we can even find practitioners saying that each day spent on requirements makes the project another day late and, whereas each day spent on coding is a day closer to meeting the deadline.” [14] The paper asks the question of how we can adapt requirements engineering to an environment that is more extreme and less controlled.
This paper stresses the need to determine which requirements engineering activities are essential to do even when the schedule is tight, and points to the possibility of having a results driven approach to requirements management. Any recommendation for doing requirements engineering under time constraints should “parameterize the circumstances of applicability in terms of such things as system size, team size, legacy vs. new development, product family vs. one of a kind, waterfall vs. prototyping, centralized vs. distributed application and development, and other characteristics of application domain.”[15] These are exactly the type of issues NASA could be thinking about, because, while some of their larger human rated projects like Shuttle and Station will eventually benefit from the CMM practices that are being introduced, the smaller projects may not. Projects that are smaller and tight on people and time could spend some time thinking about the type of project that is being carried out, and which types of requirements engineering practices are the most beneficial for their situation, before the project begins. My research has pointed out the need to have a complete understanding of the customer requirements for the software, so that the software can be adequately tested. It is also necessary to be able to do this in an environment where several subcontractors are involved. Therefore, it would be good to have more research directed towards finding out what requirements engineering and organizational practices best produce these results. It would also be good to have more research directed at discovering what experts think are the most important requirements management practices to have for different types of projects, taking into consideration characteristics such as project size, application, and number of different groups involved in the project.

In “Requirements Engineering: The State of the Practice” it is stated how “little contemporary data exists to document actual practices of software professionals for software
requirements elicitation, requirements specification document development, and specification validation." [16] This research team created a web-based survey to get real data on the state of requirements practices and aimed the survey towards industrial professionals. This survey is able to produce statistical data on the type of lifecycle used, the techniques used for elicitation and modeling requirements, the formality of the modeling notation, and the percentage of people in a certain category, say technical or managerial, that agrees with a given question.

This is the first requirements engineering survey that I have seen that has produced this much actual data regarding the practices employed in the software industry. The data indicates that while formal models and object oriented techniques are not dominant, industry perception is that we are not failing as much at requirements management as many suspect, at least in the shorter duration projects. [17] This survey had respondents from various organizational industries and types.

This type of research would be the next logical step for studying how organizational processes affect requirements management within NASA. The continuation of this research would generate more data and statistical data on how different stakeholders view different organizational issues and requirements management practices for their project, and through this, provide needed insight into how organizational and political issues impact effective requirements management, and how to manage these issues. Ideally this type of research would identify what types of requirement management practices and tools are the most essential to a specific type of project.
Notes


References


