

THE IMPACT OF ORGANIZATIONAL STRUCTURE ON FLIGHT SOFTWARE COST RISK

Jairus M. Hihn

Karen Lum

Erik Monson

*Jet Propulsion Laboratory¹
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109*

Abstract— The Jet Propulsion Laboratory (JPL) has a long record of successful deep space missions from Explorer to Voyager, to Mars Pathfinder, to Galileo to Mars Odyssey, to name but a few. Our experience and success as with the rest of the aerospace industry is built upon our hardware and system level expertise. Throughout the nineties software became more important in its contribution to spacecraft risk, integration and overall workforce. During the late nineties, this change was magnified when a number of missions managed by JPL experienced significant flight software cost growth. In addition, several of the missions have exhibited software-related schedule slips that impacted or threatened the planned launch dates. This occurred in software developed in-house as well as those that were contracted. In response JPL funded a study in 1999 to identify the systemic causes of reported flight software cost growth and to develop a set of recommendations to reduce the cost risk in flight software development activities.

The results of the 1999 study were reported in [1], [2]. In 2003, a follow-up study was conducted on seven current flight projects that launch from summer of 2001 to 2005, to see if anything had changed since 1999 and if any of the initial reports' recommendations had been implemented [3]. This paper summarizes the final results of the follow-up study updating the estimated software effort growth for those projects that were still under development and including an evaluation of the roles versus observed cost risk for the missions included in the original study which expands the data set to thirteen missions.

INTRODUCTION

Throughout the 1990's, software came to play an increasingly more significant role in spacecraft integration and risk, at the Jet Propulsion Laboratory (JPL) as well as at other aerospace companies. In the late 1990's, the importance of software was magnified when a number of JPL-managed missions experienced significant flight software cost growth. In addition, several missions had exhibited software-related schedule slips impacting or threatening the planned launch dates. This occurred in both in-house and contracted software development projects. In response, JPL funded a 1999 study to identify the systemic causes and to develop recommendations to reduce the flight software development cost risk. The results of the 1999 study were reported in two papers. The first paper identified the root causes of the observed flight software cost growth [1] and the second paper described a set of proposed strategies and policies to reduce software cost growth on future missions [2]. A major recommendation of these studies was changing the organizational structure of a software project so that the software manager had more responsibilities and accessible reporting relationships. In 2003, a follow-up study was conducted on seven current flight projects that launch from summer of 2001 to 2005, to see if anything had changed since 1999 and if any of the initial reports' recommendations had been implemented. The preliminary results of the follow-up study [3] indicated that having a software management team in place well before system PDR with budget and design authority did significantly reduce the likelihood of post-PDR software cost growth. The result that organizational and management structure of software projects has a significant impact on software cost growth, is consistent with Capers-Jones conclusion that "deficiencies of the project management function is a fundamental root cause of software disaster" [4]. The importance of communication was also documented in [5], which showed that the impact of software volatility on software development productivity was greatly mitigated when there was extensive internal and external communication.

¹ The research described in this abstract was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

This paper contains a summary of the final results of the 2003 follow-up study. In the final version the estimated cost growth for those missions not launched are updated and the role statements for the software managers were evaluated and included for the missions from the 1999 study in order to expand the data set and better document any changes in their roles between the studies. The expanded data set made it possible to expand the analysis to include the potential impacts on cost growth of the impact of inheritance assumptions, system heritage, and planning fidelity and to better comprehend the complex interrelationships between software tasks and the overall project.

Note that all but one of the participating projects were unable to explicitly choose to implement the recommendations from the 1999 study due to the timeframe of their development. Therefore, the focus of this study is on the extent to which the recommendations were beginning to be employed and measuring any impact on the observed software development cost growth. To provide proper context for interpreting the current study, the next section provides a summary of causes and recommendations that were identified in the 1999 cost risk study.

BACKGROUND: 1999 SOFTWARE COST RISK STUDY SUMMARY

Projects in either development or operations phases were selected for the 1999 study, based on the following criteria:

- Cost growth to exceed 20% of plan at Preliminary Design Review (PDR) in last three years, including one mission that was within budget
- One ground system project
- A mixture of in-house and system contracted projects

Data was gathered using a multiple step approach incorporating interviews, focus groups, multi-voting, and workshops:

1. Interviews
 - a. Using Protocol Analysis, an Unstructured Interview was performed to obtain self-reports on specific mission events.
 - b. A Structured Interview identified how self-reports had been categorized and identified missing
 - c. information.
2. A Focus Group / Workshop included the brainstorming of underlying causes of software cost growth, based on interim findings from initial interviews
3. Multi-voting identified top cost risk categories
4. A second Workshop reviewed and finalized JPL strategic software policy recommendations

From 24 missions, eight missions were chosen that were currently either in development or operation phases, including six flight software systems and two ground systems. The mission with no cost growth was used as a “control”, or assisted in verifying that projects with the identified cost risk factors did, indeed, exhibit higher rates of cost growth. Of the seven missions that experienced cost growth, the cost increased approximately 50% on average, with a range of 25% to 180%. Characteristics of the missions used in the study are described in Table 1.

Project	Flight / Ground	In-house vs. Contract	Phase during Study	Cost Growth > 20%
Project 1	Flight	In-house	Operations	YES
Project 2	Flight	In-house	Completed	YES
Project 3	Flight	In-house	Operations	YES
Project 4	Flight	Contract	Operations	YES
Project 5	Flight	Contract	Operations	YES
Project 6	Flight	Contract	Implementation	YES
Project 7	Ground	In-house	Implementation	NO
Project 8	Ground	In-house	Implementation	YES

Based upon a categorical data analysis [6], a number of key risk areas were identified. These areas included: Experience & Teaming, Planning, Requirements and Design, Testing, Software Inheritance, Staffing, and Tools &

Methods.

Table 2 contains a summary of the top five different risk areas identified in the study, the frequency with which each risk area was reported, and the sources of cost growth relating to each respective area. Based on the multi-voting process and focus group discussions, participants identified the most significant risk areas as Planning, Requirements & Design, and Experience & Teaming.

Table 2: Summary Results by Cost Risk Area		
Risk Area	Percent of Projects Reporting	Summary of Reported Issues
Experience and Teaming	71%	<ul style="list-style-type: none"> Management and system engineers lacked software experience Poor teaming between HW/ SW and systems/SW team Software team lacked mission experience
Planning	71%	<ul style="list-style-type: none"> Poor planning and estimation practices Planned inheritance never happened Insufficient reserves for SW Software staff not included in early planning and decision making
Requirements and Design	57%	<ul style="list-style-type: none"> Lack of good architecture and system partitioning Systems decisions made without accounting for impact on software SW requirements solidify late in the life cycle and are very volatile
Testing	71%	<ul style="list-style-type: none"> Testbeds; too few, too late, not validated, lacked capability
Software Inheritance	57%	<ul style="list-style-type: none"> Inherited code did not behave as advertised, was poorly documented, and required more modification than expected. (5 of 8 missions attempted to inherit software. Of these, 4 reported major problems.)
Tools etc.	86%	<ul style="list-style-type: none"> Poor test result analysis tools Purchased COTS tool never used. Not included in SW inheritance.
Staffing	71%	<ul style="list-style-type: none"> High turnover in software staff SW team not included in early stages of planning Integration and SW teams not available to support ATLO

During the interviews, participants were asked to provide preliminary recommendations to help mitigate cost growth risk in flight software projects based on personal ‘lessons learned’. Recommendations mentioned at least 50% of the time included requiring:

- Project managers & systems engineers to have a better understanding of software
- More detailed software planning and tracking similar to hardware
- Early presence of software, even in pre-Phase A, and be part of an integrated plan
- The software development process must deal with evolving requirements & assume that the unexpected will occur

The final step translated the initial recommendations into specific JPL software policies that could be implemented by managers of future missions and supported by JPL policies. The following recommendations were finalized during the second workshop:

Recommended JPL Organizational Policy

1. Require all projects to have a software system manager with budget authority, responsibility over flight and ground software and who reports directly to the Project Manager (as spacecraft and instrument managers do). The software system manager’s other responsibilities include:
 - Preparing software cost estimates, planning, and budgeting
 - Developing software architecture by PDR
 - Ensuring software and system architecture consistency
 - Ensuring software is considered in all design trades
 - Supporting subsystem managers in planning, development, integration, testing, operations, and maintenance
 - Coordinating operations, flight software, and ground software.
 - Determining how software will be managed within the project and integrated within the overall project implementation structure.

Recommended JPL Product Policies

Require the development of:

1. System architecture, supported by a software architecture that clearly documents an integrated hardware and software design prior to PDR.
2. Management plan that addresses software, including a risk management plan with reserve and contingency allocations based on estimated risk prior to PDR.
3. Test strategy and plan prior to PDR.

Recommended JPL Process Policies

1. Software Inheritance Review, similar to the Hardware Inheritance Review (when appropriate), prior to PDR and Critical Design Review (CDR).
2. Require that software be reviewed at the Non-Advocate Review (NAR).
3. Require that the software architectural designs be reviewed at PDR and updated at CDR.
4. Require that a Risk Management Plan be reviewed at PDR and updated at CDR.
5. Require Test Plans and status be reviewed at PDR and updated at CDR.

FOLLOWING THROUGH

Over the past four years, JPL has been redefining how software is integrated into its missions. The initial push has been to get key software positions defined and established in all projects. To date, implementing product and process policies has been approached more informally, allowing projects to deal with unique issues and software issues in a project-specific basis. This will begin to change in the next few years, as there will be required software document reviews as part of the major project milestone reviews. Changes have been made in creating two main positions: the Project Software Systems Engineer who gives voice to software issues at the project-level, and the Flight Software Manager who is responsible for all spacecraft software and interfaces with the instruments and the ground data systems. These positions are being staffed much earlier in the life-cycle than in the 1990's. Therefore, this study focuses primarily on how these positions have been implemented and, to a somewhat lesser extent, on the maturity and stability of the key software products and activities at system-level PDR, software and systems design stability and integration, risk management plan existence, and whether a software inheritance review was conducted. These are analyzed in the context of the observed cost growth and previously identified risk areas.

METHODOLOGY AND DATA SUMMARY

Methodology Comparison

The selection criteria for a project to be included in the follow up study were much simpler than in the 1999 study. In this case, a study of overall mission cost growth was being funded by the JPL Costing Office, which decided to take advantage of this opportunity to update the software cost risk study. Of the missions identified for the mission-level cost growth study, many were outsourced to contractors who did not have contacts easily available and could answer the detailed software-level questions. To compensate for this, data from several major instrument software projects were also included to expand the size of the data set. Table 3 summarizes the methodological differences between the 1999 study and this current study.

Table 3: Data and Methodology Differences		
	1999 Study	Current Study
Criteria	<ul style="list-style-type: none"> • Must have both in-house and contracted SW represented • $\geq 20\%$ growth from PDR to launch • At least 1 ground project • 1 project with no cost growth 	<ul style="list-style-type: none"> • Projects participating in Mission-level study • Projects that were near launch
Number of Projects	8	7
Type of Projects	<ul style="list-style-type: none"> • 2 Ground (2 in-house), • 6 Flight (3 in-house, 3 contracted) 	<ul style="list-style-type: none"> • 4 Flight (1 in-house, 2 contracted, 1 mixed) • 3 Instrument (3 in-house)

Data Collection Methodology

The data that was collected through interviews lasting approximately 60-90 minutes each. Two to three persons conducted the interviews: one functioned as the main interviewer, the second as a scribe, and the third as a backup, to reduce the likelihood that information could be lost or misinterpreted. Interviewers met to compare interview notes to identify discrepancies. Follow-up interviews were scheduled when further explanation was needed. Informal phone conversations and electronic mail were also used for further clarification, if necessary.

The interview forms consisted of the following questions:

- (1) Basic identification (name, current position, and project)
- (2) Budget at PMSR (the earliest milestone representing the gate between phase A and phase B)
- (3) Budget at PDR (milestone representing the gate between phase B and phase C)
- (4) Budget at completion or launch, or if not yet completed, then the estimate at completion
- (5) Description of software development and any issues or problems that arose
- (6) Various questions related to recommendations from the previous study, including roles and responsibilities of the software manager.

Although their roles may not have been software-specific, all participants in the study had extensive software experience. The interviewees held positions including Technical (cognizant) Engineer, Software Manager, Software System Engineer, and Flight Project Manager. The interviewees typically supported their responses to the questions with descriptions of specific events or behaviors illustrating their issues or concerns.

After the interviews were completed and transcribed, the responses were reviewed and systematically grouped into the risk areas identified in the 1999 study, as well as any new risk areas identified in the current study. Based on their findings, a table of the causes of flight software growth was constructed, followed by an assessment of in the study to determine how many of the recommendations from the previous study were implemented.

Data Summary

The current study examines software development cost on seven current JPL projects (four flight projects and three instrument projects) that have launched or have completed CDR. Three out of four flight software projects were contracted out or were partially contracted out. All three instrument software projects were developed in-house. Table 4 provides an overview of the projects included in this study and summarizes their basic characteristics. See Table 1 for comparisons to the 1999 study.

Table 4. Data Summary			
Project	Type	In-house vs. Contract	Status at Collection
Project A	Flight	Contract	In development
Project B	Flight	In-house	Completed
Project C	Flight	Mix	In development
Project D	Flight	Contract	Completed
Project E	Instrument	In-house	In development
Project F	Instrument	In-house	In development
Project G	Instrument	In-house	In development

Table 5 presents a summary of the cost growth of the projects included in this study, compared to the cost growth in the 1999 study. Average software cost growth from PDR to Launch has not changed significantly (from 51% to 53%) since the 1999 study. Mean software cost growth in Table 5 excludes growth due to major external factors. As an example, one of the projects included in the study was seriously impacted by Mars 98 mission failures and, ultimately, had to relinquish its launch opportunity to another mission (not included in this study) having a tighter launch window. Although the range of growth appears smaller, there were too few projects in the study for the range to be significant. It is important to note that unlike the 1999 study, software systems included in the study were not selected because they exhibited cost growth, so there is no a priori reason to expect an upward bias. Furthermore, this is likely to be an underestimate as some of the projects are not completed yet and may grow more than estimated.

Table 5. SW Cost Growth (Percent of SW Budget from PDR to Launch) 1999 Study vs. Current Study			
1999 Study (including Ground Software)		Current Study (all projects)	
Mean	Range	Mean	Range
51%	0-180%	53%*	8-100%

*Excludes the percentage growth due to external factors; mean would be 57% if external factors were included

RESULTS AND ANALYSIS

Understanding Cost Growth and its Sources

The reported causes of cost growth were mapped into the risk areas identified in the original study: Experience and Teaming, Planning, Requirements and Design, Testing, Software Inheritance, Staffing, and Tools. Table 6 provides a summary of the issues reported along with the corresponding frequencies in each risk area.

The results in Table 6 indicate that the key causes of software cost growth have not changed since the previous study. The issues reported are very similar and frequently identical to those reported in 1999. In all but one risk area, frequency is off by plus or minus one response between the two studies. The only exception to this is the 'tools and methods area'. In the 1999 study, this was the most frequently reported risk area but in the current study it is the least reported. One interpretation is that in the 1990's, under the severe budget pressure of Faster, Better, Cheaper, software engineers frequently went looking for a 'silver bullet', assuming that it would help reduce cost. However, it often did not work and even increased cost development cost. Today, there appears to be more concern with using basic, well-known, and mature tools in order to get the job done.

Table 6 reveals that Planning is the most frequently occurring risk area, with almost all projects in the study reporting planning issues. Planning, one of the top risk areas identified in the 1999 study, had 71% occurrence. Planning issues included: having insufficient reserves or resources for software, incorrect scoping, and poor estimation planning practices, e.g., optimistic assumptions, overestimation of productivity, and shortened Formulation Phase schedule.

Requirements and Design area is the next most frequently identified risk. Only one project out of seven had unstable

software architecture. However, five out of the other six projects reporting stable software architecture still experienced requirements volatility, improper design documentation, or lack of hardware/software architectures integration.

Table 6. Reported Risk Area Frequency with Summary Details			
Risk Area	1999 Study Percentage of Projects Reporting Responses in Risk Area	Current Study Percentage of Projects Reporting Responses in Risk Area	Summary of Reported Issues from Current Study
Experience & Teaming	71%	57%	<ul style="list-style-type: none"> • Poor teaming between HW/SW and systems/SW team • Weak communication between project mgmt and software team
Planning	71%	86%	<ul style="list-style-type: none"> • Insufficient reserves or resources for SW • Poor planning and estimation practices - optimistic assumptions, overestimate productivity, short phase A or B • Scoped incorrectly
Requirements and Design	57%	86%	<ul style="list-style-type: none"> • SW requirements solidify late in the life cycle and are very volatile • Design not fully documented/ not properly CM'ed • HW/SW architecture not integrated • Requirements immature/not well defined/not baselined • New system and software architecture
Testing	71%	57%	<ul style="list-style-type: none"> • Testbeds late/unreliable • Testbeds only had partial functionality required
Software Inheritance	57%	43%	<ul style="list-style-type: none"> • Planned inheritance was less than expected • Inherited code not the same class as other code • Inherited code was treated as if it were new code due to poor documentation
Tools/Methods	86%	29%	<ul style="list-style-type: none"> • Test tools late • Test tools lacked functionality
Staffing	71%	71%	<ul style="list-style-type: none"> • Loss of staff to other projects/High turnover in software staff; training new people takes time • Insufficient workforce • Funding profile forced us to release team and then attempt to rehire at a later date.

Staffing is the next frequently identified risk area. Staffing, a commonly identified risk area in the 1999 study, remains a highly identified risk area in this study. However, the specific staffing issues have changed. While in the 1990s, integration and software teams were not available to support ATLO (an issue reported multiple times); however, this is not an issue in this study's projects. This issue of shortage or loss of staff to other projects was reported by many projects. Participants expressed their concern that training new people consumed more time and money when the turnover is high.

Risk areas that had similar frequencies to the 1999 study include experience and teaming, testing, and software inheritance. Although the issue of poor communication existed between software teams and the rest of the flight project, experience does not appear to be an issue, as it was for the projects in the 1999 study. While the same testing issues arose in this study (as in the 1999 study), projects that experienced late testbeds reported that once the testbeds arrived, they had adequate access. Four out of seven projects in this study had software inheritance, of which four projects reported issues with their software inheritance. The project that did not have issues with software inheritance was the only project to hold software inheritance reviews.

The cost growth summary in Table 7 breaks down the results (flight versus instrument) and by growth from PMSR versus PDR. Results suggest that instrument software has higher average cost growth than flight software. Furthermore, it appears that flight software has a smaller growth range than instrument software. Yet given the small amount of data and that several projects are not yet completed, this may not be significant.

Table 7. SW Cost Growth (Percent of SW Budget) Flight Software vs. Instrument Software				
	Flight Software		Instrument Software	
	Mean	Range	Mean	Range
PMSR - Launch	69%(62%*)	23-84%	81%	52-100%
PMSR - PDR	17%	0-67%	14%	0-42%
PDR - Launch	51%(43%*)	10-80%	67%	8-100%

*Adjusted mean excludes growth due to externally caused launch slip

Table 8 presents a breakout of cost growth by project. It shows that two out of seven projects show any change in their estimate by PDR (Projects C and G). In both cases, these two projects have also experienced smaller cost growth from PDR to launch than the other five projects.

Table 8. Software Cost Growth Summary			
Project	PMSR-Launch Growth	PMSR-PDR Growth	PDR-Launch Growth
Project A	60%	0%	60%
Project B	80%	0%	80%
Project C	84%	67%	10%
Project D	23%	0%	23%
Project E	92%	0%	92%
Project F	100%	0%	100%
Project G	52%	42%	8%

This raises the question: what makes these two projects different? In both of these cases, significant attention was paid to software prior to PDR on both projects. As a result, the projects were able to identify that the PMSR budgets were underestimated, communicate this problem to the project, and finally adjust the budget accordingly.

Is The Way JPL Builds Software Changing?

During the 1999 study, virtually all flight software was developed in an unintegrated manner, under its respective hardware-oriented subsystems. As a result, the software cognizant engineers lacked budget authority and did not even have a separate account. Over the past three to four years, there has been a shift in emphasis to create higher-visibility software positions with greater authority. The current study probed to determine what software positions actually existed on the project and to what extent they were able to fulfill the recommended job role. The following nine recommended job roles are displayed in Table 9.

Table 9. Recommended Software Manager Responsibilities	
Management Responsibilities	
1.	Budget authority
2.	Preparation of software cost estimates, plan, and budget
3.	Determine how software will be managed within the project and integrated within the overall project implementation structure.
4.	Supporting subsystem managers in planning, development, integration, test, operations, and maintenance.
5.	Coordination of operations, flight software, and ground software.
6.	Manage Ground Software
Design Responsibilities	
7.	Development of software architecture by PDR
8.	Ensure consistency of software architecture and the system architecture
9.	Ensure that software is considered in all design trades

Table 10 displays a summary of responses to the survey questions that related to the implementation of the recommendations documented in the original study, which, in different forms, have been advocated by various JPL senior managers. Note that the instrument projects have not been asked to follow these recommendations presently. These instrument projects are included to evaluate what extent they fulfill the recommendations informally and to determine if underlying causes of cost growth are the same across flight systems and instrument software.

Table 10: Summary of Project Implementation of Software Recommendation						
Project	Percent Cost Growth PDR-Launch	Software Manager	Project Software Systems Engineer	Percent of Recommended Role Performed by SW Mgr and/or PSSE	Reqs/Design Stable at PDR	Software Risks identified and documented at PDR
Flight Projects						
Project A	60%	At contractor	Yes. But insufficient authority.	56%	Arch. Yes Reqs. No	No
Project B	80%	Not at SC SS level and authority diffused over several people.	Yes but insufficient authority	56%	Major Arch. elements not stable. Reqs No	Partial
Project C	9.7%	JPL SW Mgr had budget authority.	Function fulfilled by JPL SW Mgr.	83%	Arch. Yes Reqs. No	Yes
Project D	23%	At contractor	Function partially fulfilled by JPL SE and contractor SW Mgr	67%	Arch and Reqs Stable	Yes
Instrument Projects						
Project E	92%	SW CogE at too low of a level and no budget authority	Function partially performed by JPL SW CogE	67%	Partial	Partial
Project F	100%	SW CogE at too low of a level and no budget authority	Function partially performed by JPL SW CogE	56%	Arch. Yes Reqs. No	Yes
Project G	8%	SW CogE at too low of a level and no budget authority	Function partially performed by JPL SW CogE	56%	Arch. Stable, but not well integrated	No

Percentage cost growth from PDR to Launch is included to illustrate differences in cost growth between projects. The next two columns summarize whether a software manager and/or project software systems engineer (PSSE) existed on the project, whether they existed at the appropriate level, and/or whether they had sufficient authority to do the job. Three out of seven of the projects have a software manager with budget authority who reports directly to the flight system manager or instrument manager. However, only one of these projects has such a manager at JPL. The other two projects that employ a flight system manager or instrument manager are at the contractor, Project C. Furthermore, three of seven projects have someone fulfilling the project software systems engineer (PSSE) role. In Project C, the software manager fulfilled this role (found to be very effective and based on cost growth rate and percentage of recommended role). The Projects A and B the PSSE primarily fulfilled a review and advisory role, making it difficult for them to be as effective. The fourth column shows the percentage of the recommended functions that were performed by the software manager and/or PSSE (see Table 9). No project has implemented all of these functions as originally recommended, i.e., none of the studies in the survey had an integrated flight-ground software manager. However, all four of the flight software systems (and even the three instrument projects included in the study) implemented at least 50% of the recommendations. The last two columns indicate if the software requirements and architecture were stable by PDR and if the software risks were formally documented at the project level or in a software risk management plan. As before, software requirements are not stable at PDR - which will likely always be the case for JPL given the nature of its business. Significant progress has been made in establishing a stable software architecture by PDR, however. While there is greater inconsistency, progress is also being made in identifying and tracking software risks and in establishing a software management plan.

Table 10 illustrates that out of all projects included in the study, Project C was the only project with a JPL Flight Software Manager having both budget and technical authority. Project C scored the highest with 7.5 out of nine (83%) of the recommended roles being fulfilled. A half point was given to a partially fulfilled role. Of the Flight

Projects, Project C also has the lowest cost growth from PDR. (Refer to Figure 1 for this illustration, below). Clearly, there are other differences between these projects as well as similarities besides the differences in the role of the software manager. The major difference is that Project B is larger and more complex than Project C. However, Projects B and C have similar cost at complete and Projects B and C also have similar cost growth from PMSR. While this one observation is not sufficient to prove our hypothesis, it is consistent with the conclusion that the recommendations from the previous study can reduce software cost growth, as illustrated by Figure 1.

Although the participants in the original 1999 study were not asked about their management roles directly, the percentage of the recommended roles performed by the software manager or project software system engineer was estimated by revisiting the original survey forms. Growth rates from PMSR were not available. The equivalent percent of recommended roles performed and other summary data for projects in the 1999 study are presented in Table 11. Not surprisingly there were no PSSE's as the position had not been created yet and the roles fulfilled are similar to the projects shown in Table 10 mostly ranging from 56% to 61%.

In the preliminary study, the four flight software datapoints suggest a relationship between cost growth and role percentage. Combining the 1999 study results with the data from this follow-up study allowed us to perform an ordinary least squares regression on cost growth versus role percentage, which did not yield any significant results. However, the additional data makes it possible to at least consider the impact of more than just the roles fulfilled by the software management team on cost growth. A variety of factors were considered. A significant relationship was found between percentage of cost growth and the combination of percentage of software roles fulfilled and the occurrence of overly optimistic software inheritance assumptions. A significant relationship was found between percentage of cost growth and the combination of percentage of software roles fulfilled and the occurrence of overly optimistic software inheritance assumptions. This result is displayed in Figure 1.

Project	Percent Cost Growth PDR-Launch	Software Manager	Project Software Systems Engineer	Percent of Recommended Role Performed by SW Mgr and/or PSSE
Project 1	30%	Multiple SW CogE's at lower level , no budget authority	Position did not exist Function partially fulfilled by multiple SW CogE's	56%
Project 2	80%	Single SW CogE but no budget authority and at lower level	Position did not exist Function mostly fulfilled by JPL SW CogE	61%
Project 3	35%	Mutliple CogE's at JPL and Contractor at lower level; some had budget authority	Position did not exist Function partially fulfilled by multiple SW CogE's	61%
Project 4	60%	At contractor only	Position did not exist Function partially fulfilled by JPL SE and contractor SW Mgr but too distributed	67%
Project 5	55%	At contractor only	Position did not exist Function partially fulfilled by JPL SE and contractor SW Mgr but too distributed	56%
Project 6	120%	At contractor only	Position did not exist Function partially fulfilled by JPL SE and contractor SW Mgr but too distributed	56%

Figure 1 shows flight software cost growth versus percent of software roles fulfilled for the flight projects from the 1999 study together with the flight projects in this study. Projects B, 4, 5, and 6 were projects in which the project participants indicated that the original inheritance assumptions were overly optimistic. It appears that a strong relationship to cost growth, with an adjusted R-square of .63, exists between inheritance and role percentage. Inheritance was treated as a qualitative variable with a value of 1 for overly optimistic inheritance assumptions, and a value of 0 for realistic inheritance assumptions. A multiple regression with role percentage and inheritance as

independent variables appears to be statistically valid with an F-value of 9 at an observed significance level of .05. All t-statistics are significant at the ten-percent level. These results indicate that there is a statistically significant relationship between percent of software roles fulfilled and flight software cost growth when inheritance is taken into consideration (Figure 1).

It is the combination of these two factors – software role percentage and overoptimistic inheritance assumptions – that gave us these significant results. While there are other factors that can cause cost growth, there is insufficient evidence to support the relationship of cost growth to other contributors of cost growth. There was no significant relationship found between cost growth and whether the software project was contracted or in-house. Another dimension – newness of the technology or precedentedness – was also analyzed. There was insufficient evidence to prove a relationship between the precedentedness of the software and the percent of software cost growth. There was also no software cost growth pattern found over time when the data is laid out by project launch date.

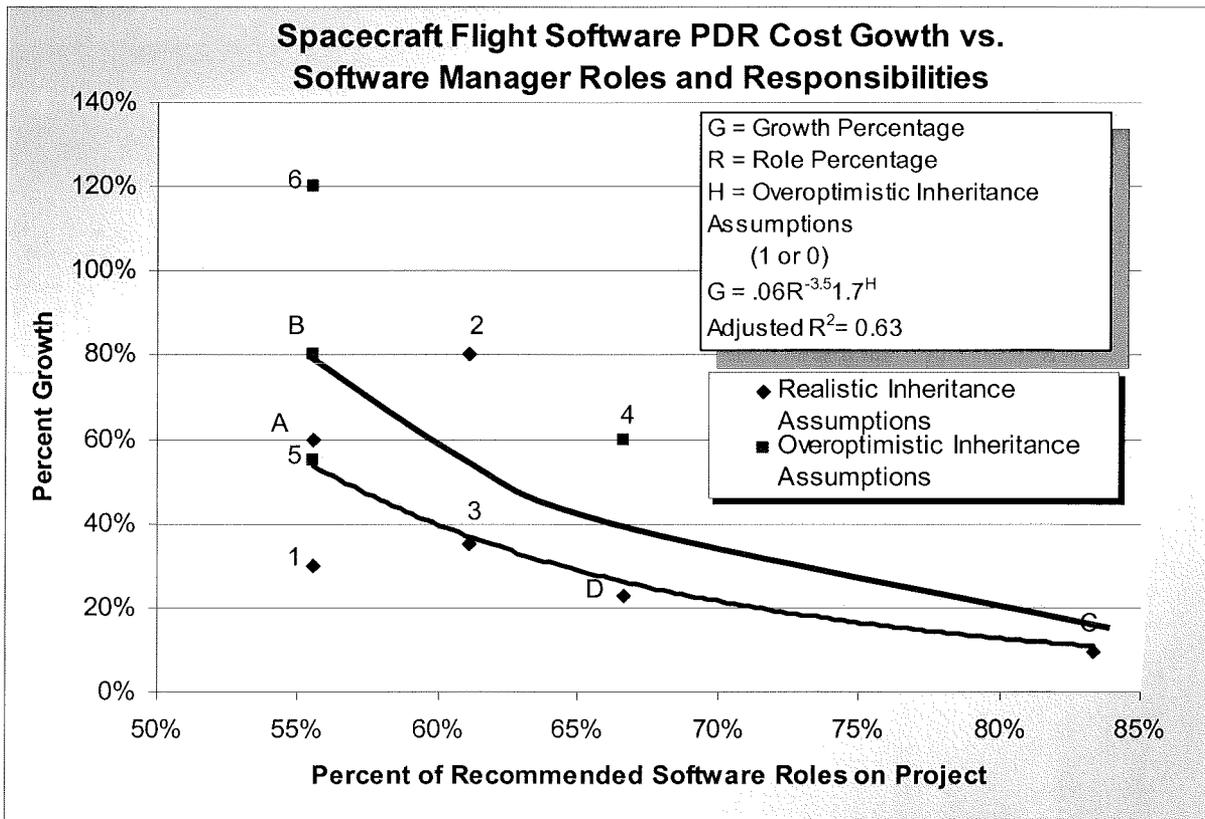


Figure 1 Spacecraft Flight Software Cost Growth vs. Software Manager Roles and Responsibilities

CONCLUSIONS

The purpose of the study was to document whether the recommended Flight Software Manager roles and responsibilities as documented in the 1999 cost risk study may have been implemented and whether they had any effect in reducing software cost growth. A secondary objective was to document any trends in software cost growth. There are, however, a few important considerations to be mentioned. The sample size is relatively small and only one of the seven projects in the follow-up study had time to formally implement the recommendations.

It was found that average cost growth had not significantly changed since the 1999 study. Furthermore, the underlying causes of such cost growth were fundamentally the same as found previously. Clearly, increased effort is required to catch cost growth prior to PDR in order to eliminate extensive software cost growth after the commitment review. With respect to the Flight Software Manager recommendations, it was found that all seven projects

implemented at least 50% of the recommended roles between their Software Manager and/or Project Software Systems Engineer. While more data is needed, results from this follow-up study support the importance of software 'visibility' at the project-level and the delegation of responsibility to a strong software manager with budget and technical authority. In order to be effective, these positions also need to be filled well in advance of PDR. With the combined dataset from the 1999 study and this current study, the main contributors to flight software cost growth are the combination of percent of software roles fulfilled and overoptimistic inheritance assumptions.

These results are consistent with the 1999 study recommendations that a strong, highly visible software management team at the project- and system-level is needed. They need to have budget authority to make decisions in a timely manner, and they need to have major input into system-level design decisions. The results support the assertion that greater discussion and visibility of software earlier in the lifecycle will help managers to identify and address the most troublesome problems before they lead to significant cost and schedule growth.

REFERENCES

- [1] Hihn, J and Habib-agahi, H. "Identification and Measurement of the Sources of Flight Software Cost Growth", *Proceedings of the 22nd Annual Conference of the International Society of Parametric Analysts (ISPA)*, 8-10 May 2000, Noordwijk, Netherlands.
- [2] Hihn, J and Habib-agahi, H. "Reducing Flight Software Development Cost Risk: Analysis and Recommendations", 2000-5349, *Proceedings AIAA Space 2000*, 19-21 September 2000, Long Beach, CA.
- [3] Hihn, J, Lum, K, Habib-agahi H., and Monson, E. "Managing Flight Software Cost Risk", *Proceedings AIAA Space 2003*, 23-25 September 2003, Long Beach, CA.
- [4] Jones, Capers. *Patterns of Software System Failure and Success*. International Thomson Computer Press, 1996.
- [5] Hihn, J., Malhotra, S. and Malhotra, M.. "Volatility and Organizational Structure" *Journal of Parametrics*. September 1990. pp. 65-82.
- [6] Simon H. and Ericson, K., *Protocol Analysis: Verbal Reports as Data*, MIT Press, 1993.

Jairus Hihn has a Ph.D. in Economics with principle application areas in econometrics and mathematical economics. He has been developing estimation models and providing software and mission level cost estimation support to JPL's Deep Space Network and flight projects for the past fifteen years, Jairus is currently the lead for the Software Quality Improvement Project's Measurement and Estimation (M&E) Element, which is establishing a laboratory wide software metrics and software estimation program at JPL. M&E's objective is to enable the emergence of a quantitative software management culture at JPL.

In a previous incarnation, Dr. Hihn was on the Faculty at UC Berkeley in the Department of Agricultural and Resource Economics where he co-developed a new statistical technique based on the semi variance of a probability distribution for use in estimating agricultural production and income risks; was the co-author on several papers which formally applied catastrophe theory to the analysis of political instability in third world countries using both non-parametric and maximum likelihood methods. He has extensive experience in simulation and Monte Carlo methods with applications in the areas of decision analysis, institutional change, R&D project selection cost modeling, and process models.

Karen Lum is involved in the collection of software metrics and the development of software cost estimating relationships at the Jet Propulsion Laboratory. She has a MBA in Business Economics and a Certificate in Advanced Information Systems from the California State University, Los Angeles. She has a BA in Economics and Psychology from the University of California at Berkeley. She is one of the main authors of the JPL Software Cost Estimation Handbook. Publications include Best Conference Paper for ISPA 2002: "Validation of Spacecraft Software Cost Estimation Models for Flight and Ground Systems."

Erik Monson is involved in supporting cost estimating activities at the Jet Propulsion Laboratory. He holds a Master of Business Administration degree from Claremont Graduate University and a BS in Computer Information Science from Lock Haven University. Previously, Erik was the lead software developer for a Silicon Valley startup where he developed an application using network-layer packet sniffers and automated knowledge bases to track inappropriate usage on large distributed networks.