Final copy

Identification and Estimation of Flight Software Cost Risk Growth

Jairus M. Hihn, Ph.D.
Hamid Habib-agahi, Ph.D.

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, Ca. 91109

1.0 INTRODUCTION

The Jet Propulsion Laboratory (JPL) in Pasadena, California is a US Government Federally-Funded Research and Development Center that is run by the California Institute of Technology for the National Aeronautics and Space Administration (NASA)[1] JPL's primary role is to conduct unmanned, robotic space exploration missions throughout our solar system. JPL has a long record of successful deep space missions from Explorer to Voyager, to Mars Pathfinder. Our experience and success as with the rest of the aerospace industry is built upon our hardware and system level expertise. The majority of JPL's, as well as other aerospace organizations, current managers and system engineers have made their reputations on these hardware intensive spacecraft. Only recently has software become more important in its contribution to spacecraft risk, integration and labor cost, therefor more emphasis on software development management and planning is required. This paper reports the results of a study funded by JPL's Develop New Products (DNP) reengineering project. The study objectives were to;

- Identify the high level sources of the unexpected software development cost growth and quantify their impact,
- Incorporate the study results into a high level parametric software cost model.

2.0 BACKGROUND

JPL, along with the rest of NASA and industry, is actively engaging the "faster, better, cheaper" philosophy. The first "faster, better, cheaper" mission was Mars Pathfinder. It was 70% less than the average cost of JPL's major space missions from 1964 to 1994. In addition, the frequency of missions launched has dramatically increased, from the historical rate of about one mission every two years to multiple missions launched every year. This shift has required major changes in the way JPL does business and is creating some institutional strain as the organization finds ways to adapt.

In response, JPL is undergoing a radical redefining of its development processes to provide future missions the means to achieve the goals of "faster, better, cheaper". The focus of these activities until very recently has been primarily on improving the development process and tools for hardware employing concurrent engineering and information technology. Also underway is what appears to be a very successful shift towards incorporating off the shelf hardware for many mission components. However, during this same time of seeking new ways of doing business, spacecraft have become more software intensive. This paper addresses the causes of flight software cost growth and quantifies their

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

impact. The results can be used to help managers identify and estimate the main cost risk drivers, which can be used to determine what changes to make in order to reduce the overall cost and to develop a more accurate software cost estimate.

3.0 METHODOLOGY

3.1 Sample Definition

In order to limit study cost and time, it was decided not to make an exhaustive or stratified survey of all flight software under JPL management.

Missions were included in the study based on the following criteria:

- Cost growth had to exceed 20% of plan at PDR in the last 3 years, except one mission had to be included that stayed within its budget
- At least one ground support software development task had to be included
- A mixture of in-house and subcontracted missions had to be included.
- Participants were chosen based on having worked extensively on the selected missions

3.2 Data Collection Methodology

The data collection methodology was relatively complex consisting of multiple steps:

1) Unstructured Interview based on Protocol Analysis to obtain self reports of what happened on specific missions
2) Follow up Structured Interview to verify how self reports had been categorized and to identify missing information

3.3 Interviews

The two interview sessions consisted of approximately 60-90 minutes each. The interviews primarily focused on a single selected mission. Two to three persons conducted both interviews. One interviewer functioned as the main scribe and interviewer, the others as backup to reduce the likelihood that information was lost or a potentially important point was missed. The approach used is a modification of the Protocol Analysis methodology proposed by Simon and Ericson[2]. It translates self-reports into ordinal data by grouping descriptive information into categories.

The first interview was relatively unstructured and consisted of only four basic questions. The open ended interview questions were designed to elicit information concerning what was working and not working within the selected mission/project. Detailed notes were taken from each interview, which were transcribed and used as the information source from which data could be derived. The interviewees typically responded to the questions by describing specific events or behaviors that supported their response and illustrated their issues or concerns.

The questions for the first interview consisted of:

(1) Identify mission/project, mission objectives, role

(2) Describe the main causes of the software development cost growth experienced on your project

(3) Identify the top three software development cost risks based on your experience

(4) Describe what you will do differently, to reduce the software development cost risks you identified

[2] Simon, H. and Ericson, K., Protocol Analysis; Verbal Reports as Data, MIT Press, 1993

After the interviews were completed and transcribed, the responses were reviewed and systematically grouped into common themes, which resulted in the identification of 7 major cost growth risk areas or categories. After developing a first draft set of causes and potential recommendations, a second interview with the same group was initiated in order to review how their information had been mapped into the identified cost growth categories. The respondents could modify the information, add new information, delete information and even add new categories. In addition, the participants were asked to provide a subjective estimate of the overall cost growth as a percentage of the budget/plan at the time of the Preliminary Design Review (PDR) and the percentage contribution of each cost growth category. The tables were updated based on the information provided by the participants. The analysis was performed based on this information. An example of a cost risk table from an actual interview is displayed in Figure 1.

Risk Area	**Percentage contribution to cost growth**	**Participant Statements Grouped by Risk Area**
Experience & Teaming	10%	• System engineers had extensive HW experience but had limited SW experience • Managers were HW oriented and had limited understanding of SW
Planning, Estimation & Control	50%	• Software development cost was underestimated because (1) had assumed too much inheritance and (2) had underscoped effort partly because had not accounted for code growth.
Requirements & Design	10%	• Did not have sufficient traceability between system and SW requirements and also had no process to feed back information as learned more about SW requirements.
Testing	15%	• Not enough testbeds • Simulators were not ready until late in lifecycle, which delayed testing • Testbeds functional capability were limited
Software Inheritance	included in planning	• Assumed software inheritance of 30% from a previous mission but ended up inheriting less than 5%.
Tools & Methods	10%	• Missed some test results because analysis tool lacked capability
Staffing	5%	• Persistently under staffed.
TOTAL Cost Growth	50%	• Project completed with 50% cost growth in flight software development costs

Figure 1: Example Cost Risk Table

4.0 DATA SUMMARY

A total of 11 managers and engineers provided information for this study. They held a variety of positions from Technical (Cognizant Engineer) to Project Manager. The breakdown is displayed in Figure 2. Seven of the participants had extensive software experience and 6 had limited experience.

Position	No. of Participants
Project Manager	3
Spacecraft Manager	3
Technical Manager	3
Systems Engineer	1
Software Manager	1
TOTAL	11

Figure 2: Distribution of Participant Positions

The study included software from 8 missions out of 24 that are currently either in development or operations. An overview of the missions included in the study and their basic characteristics is presented in Figure 3. There are six flight and two ground systems with some having completed implementation and some still under development; three were subcontracted; and only one of the missions included in the study has not exhibited any cost growth.

Mission	Flight/ Ground	In-house vs. Contract	Current Phase	Cost Growth > 20%	Number of Participants
Mission 1	Flight	In-house	Operations	Yes	2
Mission 2	Flight	In-house	Completed	Yes	2
Mission 3	Flight	In-house	Operations	Yes	1
Mission 4	Flight	Contract	Operations	Yes	1
Mission 5	Flight	Contract	Operations	Yes	1
Mission 6	Flight	Contract	Implementation	Yes	2
Mission 7	Ground	In-house	Implementation	No	1
Mission 8	Ground	In-house	Implementation	Yes	1

Figure 3: Summary of Mission Characteristics

SW Cost Growth+ (Percent of SW Budget)	
Mean	Adj Range
51%	25%-71%

Figure 4: Software Cost Growth Summary

A summary of the cost growth of the missions included in the study is provided in Figure 4. This excludes the missions with lowest and highest observed cost growth. Three of these were based on actual data and four were based on a combination of recollection and data. This shows that the average cost increase for projects that experience cost growth, is approximately 50% with a range of 25% to 75%. Of course, one should not conclude that all flight software developments would exceed their PDR plan by 50% based on the results of this study. The sample size is small and the flight software tasks were

+ Summary excludes the highest and lowest observations. The mean with all observations is 54%.

preselected based on the condition that they had more than 20% cost growth over the plan at PDR. How to actually interpret the data will be discussed in the next section.

5.0 RISK IDENTIFICATION AND QUANTIFICATION

5.1 High Level Risk Identification and Quantification

Based on a categorical analysis of the data a number of key risk areas were identified. These areas are: Experience & Teaming, Planning, Requirements and Design, Testing, Software Inheritance, Staffing, and Tools & Methods. Figure 6 provides a summary of the percentage of missions reporting issues associated with each risk area.

As Figure 5 shows, no single risk area was reported by all missions. However, most of the missions did report cost growth arising from 5 out of the 7 risk areas. The most frequently identified risk area was Tools & Methods, which came up in 86% of the missions. Several missions reported a lack of test analysis tools that in one case caused an important anomaly, which was not resolved until just before launch. There are also instances where COTS tools were purchased and never used because they turned out to be inadequate. In one case the cost of a COTS tool, which was not used, was equivalent to 5% of the total cost growth.

Risk Area	Percentage of Missions Reporting Responses in Risk Area	Summary of Reported Issues
Experience & Teaming	71%	• Management and system engineers had insufficient software experience • Weak teaming between hardware, software and systems teams • SW engineers lacked system and mission experience
Planning	71%	• Planning and estimation practices • Planned inheritance never happened • Insufficient reserves for SW
Requirements & Design	57%	• Lack of good system architecture and system partitioning • Lack of good software architecture • Systems decisions made without accounting for impact on software • SW requirements solidify late in the life cycle and are very volatile
Testing	71%	• Testbeds; too few, too late, not validated, insufficient capability
Software Inheritance	57%	• Inherited code did not behave as advertised and required more modification than expected. (5 of 8 missions attempted to inherit software. Of these, 4 reported major problems.)
Tools & Methods	86%	• Key test results were missed because analysis tool lacked capability • Purchased COTS tools that were not used.
Staffing	71%	• High turnover in software staff • SW team was not included in early stages of planning • Integration and SW teams were not available to support ATLO

Figure 5: Reported Risk Area Frequency with Summary Details (Based on missions reporting cost growth)

The next most frequently identified risk areas related to Planning, Experience & Teaming, Testing and Staffing categories. Over 70% of the missions planning problems were related to incorrect software assumptions, especially optimistic assumptions with respect to inheriting software from other missions and projects. This is further exacerbated by the impact of traditionally starting software late in the mission development process. Three of the participants discussed the concern that there is fundamentally greater uncertainty associated with software development than with hardware. The lack of sufficient software development knowledge in the project office, and a lack of systems and mission perspective on the part of software engineers magnify the inherent development risks. Also the lack of communication between hardware and software teams and, in many cases system decisions being made without considering the impact on software, has led to increased software development costs. The problems cited in Testing were: (1) not having access to multiple testbeds and simulators early enough; and (2) the lack of available staff from system integration and software development teams to support the functional testing and ATLO team (Assembly, Test, Launch, Operations).

Missions reported cost growth due to problems with Requirements & Design 57% of the time. There were a number of issues, many of which were process related, that arose in how system architecture was developed. A key problem was the lack of a well defined system architecture supported by a software architecture, which is required to provide the structure needed as the mission and system evolve over the life cycle.

The frequency of occurrence of a risk area is not the same as its actual impact, therefore, estimates of each risk areas contribution to cost growth were also obtained from the participants and are summarized below in Figure 6. The data provided is based on the participants' best judgment. The mean values were computed by assuming that ranges represented a uniform distribution which were combined using Monte Carlo techniques to compute the mean of the overall distribution. Note that software inheritance was combined with planning as several of the respondents felt it was not possible to separate it from the planning problems that arose on their mission.

RISK AREA	Range	Mean
Experience & Teaming	5-15%	10%
Planning	20-50%	35%
Requirements & Design	10-50%	25%
Staffing	5-25%	10%
Testing	10-30%	15%
Tools	5 -20%	5%

Figure 6: Risk Impacts

Based on the missions in the study, there is an indication that the two highest risk areas relate to Planning and Requirements & Design, which accounted for 60% of the observed average cost growth. The next highest risk area was Testing which reflects the impact of not having sufficient testbeds and simulators and/or their being delivered too late. Note that even though Tools & Methods issues arose with high frequency, the estimated impact on cost growth was very small.

The results only change slightly when combining risk frequency and estimated impact. Planning problems are clearly the largest source of expected cost growth for flight software. Requirements & Design and Testing are about equal in expected cost growth impact. This study would indicate that these three areas are the ones that managers need to address carefully in the early stages of the mission lifecycle in order to reduce the likelihood of software development cost growth.

6.0 SOFTWARE COST RISK MODEL

At this point there is not sufficient data to formally estimate model parameters. However, if we assume a typical log linear model similar to the original COCOMO[3] form, e.g.

$$Effort = \alpha * SLOC^{\beta}\prod_i CD_i$$

Where SLOC = source lines of code

CD_i = i th cost driver

i = 1,n

j= 1,m

Based on the subjective estimates provided by the participants it is possible to heuristically propose some potential weighting factors for the set of risk drivers.

Figure 7 presents our proposed risk driver weights for a risk model of the form:

$$Effort = \alpha * SLOC^{\beta}\prod_i CD_i \prod_j RD_j$$

RD_j = j th risk driver.

α,β are technical coefficients

i = 1,n

j= 1,m

Risk Driver	Cost Risk Driver Weights			
	Nominal	High	Very High	Extra High
Experience & Teaming	1	1.02	1.05	1.08
Planning	1	1.10	1.17	1.25
Requirements & Design	1	1.05	1.13	1.20
Staffing	1	1.02	1.05	1.13
Testing	1	1.05	1.08	1.15
Tools	1	1.02	1.03	1.10
Maximum Expected Cost Threat	0%	30%	60%	132%

Figure 7: Proposed Risk Driver Weights

The risk driver weights were derived using 50% of the low, mean and high values from the estimated risk impacts (see Figure 6). The lows were rounded down and the extra highs were rounded up. The 50% is based on the average cost growth of the mission included in the study (see Figure 4). These weights are based on the JPL environment over the past 3 years. The appropriate values will likely

[3] Boehm, B., Software Engineering Economics, Prentice Hall, 1988

change as JPL changes its software development process and methods; they are also different for different organizations. The last row, labeled Maximum Expected Cost Threat, shows the net impact of all the risk drivers; High, Very High, or Extra High, respectively.

A set of risk driver descriptions is provided in Figure 8. The descriptions are derived from the information provided in the interviews. What is described in Figure 8 are the end points of the scale, which are indicated as Nominal and Extra High. If the appropriate rating falls between the end points, i.e. High or Very High, then the ratings need to be subjectively evaluated as to the conditions that describe the task being evaluated.

A cost growth multiplier can be computed using the following steps.

1. Use Figure 8 to subjectively determine the rating of each risk driver, depending on its relative importance and variation between Nominal and Extra High
2. Use Figure 7 to determine the risk driver weight, RD_j, associated with each risk driver rating.
3. Derive the cost growth multiplier, $\prod_j RD_j$, by multiplying the risk driver weights.

The cost growth multiplier is an indicator of the cost risk that a software development project is likely to experience. The cost growth multiplier can be used to indicate the size of the reserve that should be held or to suggest by how much to increase the software budget. It can also be used to help managers identify the main risk drivers for their project and to determine what changes to make in order to reduce the overall cost risk and required reserves. These current values are based on a small sample. While derived in a reproducible manner, they are only suggestive of the likely values and a more extensive study based on actual project data needs to be performed.

Risk Driver	**Software Cost Risk Driver Ratings**	
	Nominal (Reduces Risk)	Extra High (Increases Risk)
Experience & Teaming	• Extensive software experience in the project office • Software staff included in early planning and design decisions • Integrated HW and SW teams	• Limited software experience in the project office • Software staff not included in early planning and design decisions • HW and SW teams are not integrated
Planning	• Appropriately detailed and reviewed Plan • All key parties provide input with time to get buy-in • Appropriate assignment of reserves • SW inheritance verified based on review and adequate support	• Lack of appropriate planning detail with insufficient review • Not all parties involved in plan development • Simplistic approach to reserve allocation • Optimistic non-verified assumptions especially with respect to software inheritance
Require-ments & Design	• Solid system and SW architecture with clear rules for system partitioning • Integrated systems decisions based on both HW and SW criteria • SW Development process designed to allow for evolving requirements	• System and Software architecture not in place early with unclear descriptions of basis for HW & SW partitioning of functionality. • Systems decisions made without accounting for impact on software • Expect SW requirements to solidify late in the life cycle
Staffing	• Expected turnover is low • Bring software staff on in timely fashion • Plan to keep software team in place through launch	• Expected turnover is high • Staff up software late in life cycle • Plan to release software team before ATLO
Testing	• Multiple Testbeds identified as planned deliverables and scheduled for early completion. • Separate test team • Early development of test plan	• Insufficient Testbeds/simulators dedicated to SW and are not clearly identified as project deliverables • Plan to convert SW developers into test team late in lifecycle • Test documents not due till very late in the life cycle
Tools	• CM and Test tools appropriate to task needs • Proven design tools	• No or limited capability CM and test analysis tools • Unproven design tools selected with limited time for analysis

Figure 8: Cost Risk Driver Ratings

7.0 CONCLUSIONS

A number of JPL managed missions have experienced cost growth with respect to the flight software portion of the mission. This has occurred for both in-house and subcontracted missions. The results from this study indicate that the sources of flight software cost growth can be categorized into a small number of basic risk areas with three accounting for 75% of the cost growth, Planning, Requirements & Design, and Testing. Two other key risk areas relate to Software Inheritance and Experience & Teaming. The study results also indicate that given the current software development environment and approach, that software development reserves greater than 30% are likely to be required. In addition, a basic template for a software cost risk model has been proposed with estimates for the mean impact of the risk drivers, based on the use of a methodology for obtaining subjective cost growth estimates when quantitative data is not available

The next step is to obtain quantitative data from both completed and on-going missions to provide better mission and software development information and metrics. This data can then be used to develop more detailed cost estimation models, guide managers in better planning and control practices, and also support the development improved software development process.

Acknowledgements

This work would not have been possible without the valuable assistance and close cooperation of the JPL Flight Software Cost Risk Study Team, Dave Eisenman, Dan Erickson, Rick Grammier, Chris Jones, John Lai, Rob Manning, Phil Morton, Dave Nichols, Moshe Pniel, Glenn Reeves, Larry Simmons, Fernando Tolivar and also The Center for Space Missions Information and Software Systems (CSMISS) contributors; Frank Kuykendal, Tom Fouser, Milt Lavin, and Charles Beswick. Special thanks to Dave Eisenman, Chris Jones, Moshe Pniel, Charles (Jeff) Leising, Dan Erickson, Randy Taylor and Richard O'Toole for their comprehensive review.