Software Cost Estimation for the LADEE Mission

Howard Cannon
Intelligent Systems Division
NASA-Ames Research Center
Moffett Field, CA 94035
howard.n.cannon@nasa.gov

Karen Gundy-Burlet
Intelligent Systems Division
NASA-Ames Research Center
Moffett Field, CA 94035
karen.gundy-burlet@nasa.gov

Abstract—**The purpose of the Lunar Atmosphere Dust Environment Explorer (LADEE) mission was to measure the density, composition and time variability of the lunar dust environment. The ground-support and onboard flight software for the mission was developed using a "Model-Based Software" methodology. In this technique, models of the spacecraft and flight software are developed in a graphical dynamics modeling package. Flight Software requirements are prototyped and refined using the simulated models. After the model is shown to work as desired in this simulation framework, C-code software is automatically generated from the models. The auto-generated code is then integrated with the Core Flight Executive and Core Flight Services (cFE/cFS) packages, VxWorks and appropriate board support packages. The generated software is then tested in real time Processor-in-the-Loop and Hardware-in-the-Loop test beds.**

Software cost estimation for the mission was performed 3 ways: 1) Extrapolated from development of a earth-based prototype hover-test vehicle, 2) Estimated through the Goddard Space Flight Center "mission design center" 3) Through the use of COCOMO based estimation spreadsheets. In this paper, we will discuss the characteristics of each of the cost estimation methods, and how they were tuned for a model-based development effort rather than a traditional effort. The estimates are also compared with actual costing and trend data for the LADEE Flight Software effort.

Keywords—software cost estimation, model-based software development

TABLE OF CONTENTS

I. INTRODUCTION

The Lunar Atmosphere Dust Environment Explorer (LADEE) [1] was a small NASA explorer class mission that launched September 6 of 2013 and was successfully deorbited on April 18, 2014. It orbited the moon to determine the density, composition and variability of the lunar exosphere. To minimize fabrication and design costs, the "modular common bus" spacecraft was designed with common structural components that could be connected together to form the spacecraft. LADEE was formed of four such modules, housing the propulsion, avionics and three instruments for observation of the moon.

LADEE's physical construction proceeded down a low-cost rapidly prototyped product-line effort, and a similar philosophy drove the development of the software base. With strict launch deadlines, budget limitations and a short 100-day mission plan, the flight software team surveyed successful examples of rapidly deployed small-spacecraft missions to discover best practices. Of these, the AFRL XSS-10 and XSS-11 spacecraft utilized model-based development methodologies, and we proceeded with a trade study between modeling languages using a hover test vehicle [2]. One advantage with model-based development is that controls engineers are able to rapidly prototype algorithms in their "native language" and software engineers could directly auto-code the models. This minimizes the opportunity for communication errors between algorithm designers and qualified software developers. The model based-methodology also enhances early prototyping of requirements, enables validation and verification during early stages of development and provides a common platform for communication between subsystems, software engineers and stakeholders.

One difficulty with Model Based software development is that while there were anecdotal cases like XSS-10/11, there was limited literature on how to modify cost estimation to take into account the reputed savings of the technique. An additional factor in uncertainty of the mission cost was the background of the team: experienced software, systems and GN&C researchers and engineers, but without prior experience in developing onboard flight software for spacecraft. To minimize budget risk, we researched best practices in cost estimation, and found extensive literature on cost estimation techniques [3]. We utilized three different cost estimation techniques, including model based parametric techniques, expert judgment estimation and analogy from internal prototyping efforts.

II. Software Architecture

For the LADEE mission, the flight software was developed utilizing a technique known as "Model-Based Software Development." In this technique, models of the spacecraft and high-level flight control software are developed in a graphical modeling language, such as Mathworks' Simulink [4]. Flight Software requirements are prototyped, refined and partially verified using the simulated models. After the model is shown to work as desired in this simulation framework, "C"-code software is automatically generated from the models. A Simulink Interface Layer then wraps the auto-generated code to communicate across the bus with the Core Flight Executive (cFE), Core Flight Software (cFS) and Operating System Abstraction Layer (OSAL) [5,6] GOTS components developed at NASA GSFC. These are then layered on VxWorks [7] and a Board Support Package (BSP) for the BroadReach Integrated Avionics Unit (IAU). The BSP was customized to be compliant with the Portable Operating System Interface (POSIX) standard defined by the IEEE Standards Association [8].

Figure 1 shows the integrated architecture for the LADEE mission. The yellow circles indicate control functions that were developed in Simulink and auto-coded for integration with the other elements. Blue circles indicate the tasks that were reused from the cFS package. While the cFS code was wholly reused, the tasks were customized for the LADEE spacecraft through modifications to tables and headers. The circles in green represent tasks that were newly developed hand-code, including hardware drivers for all sensors, actuators and payloads as well as commanding, telemetry and memory scrub functions. The modules communicate across the bus using the cFE "publish/subscribe" paradigm. A feature of the LADEE software is the implementation of POSIX standards within the device driver layers.

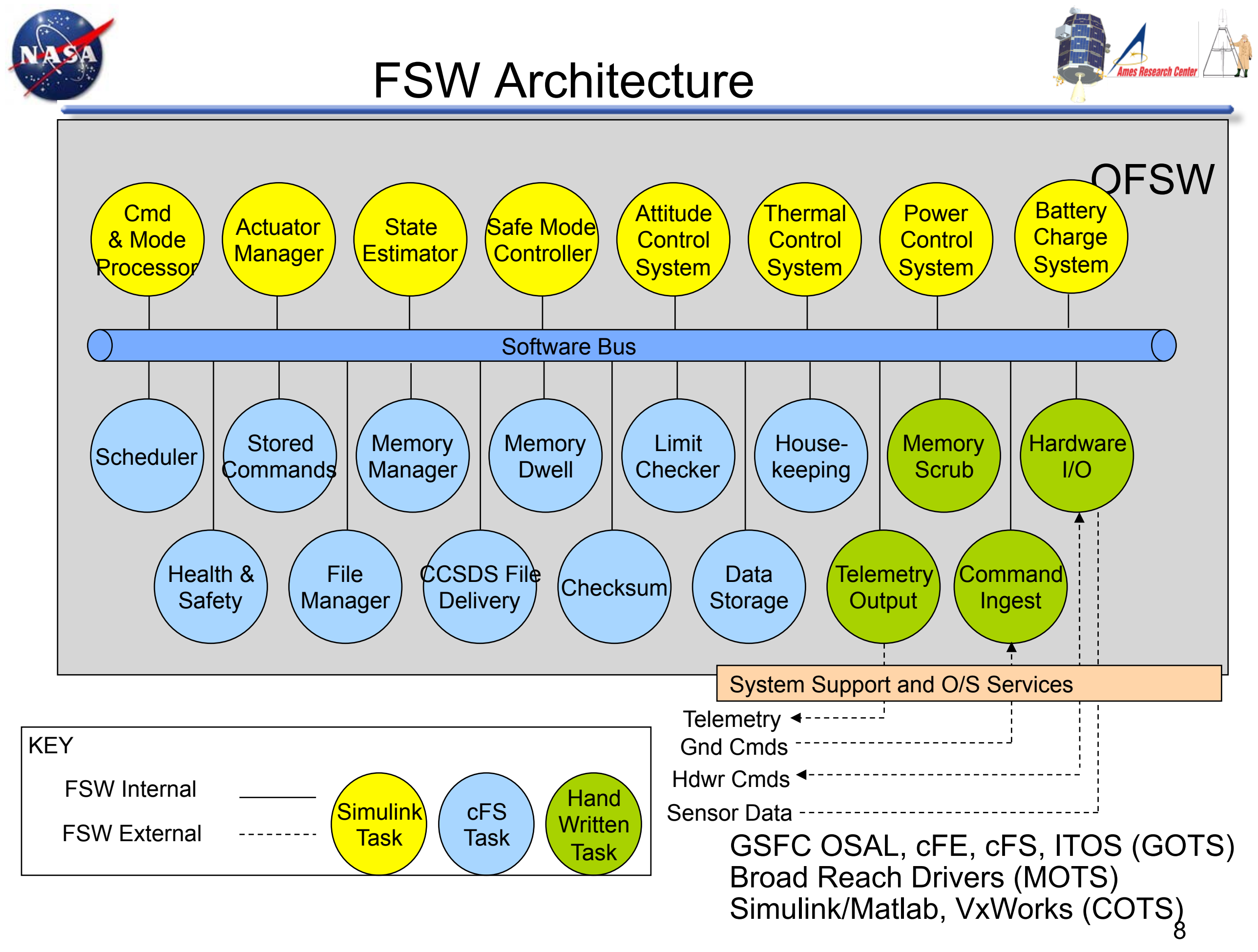


Figure 1 Flight Software Architecture

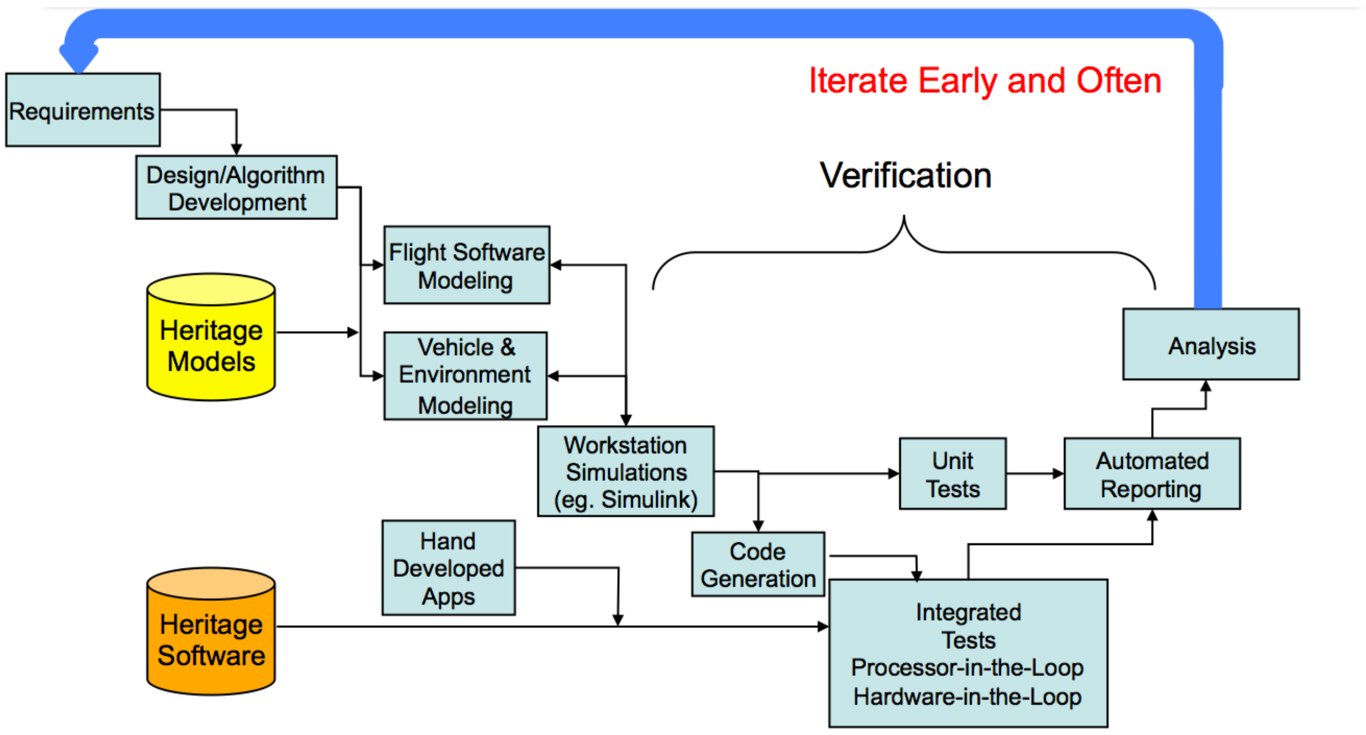


Figure 2 Agile Software Development Process

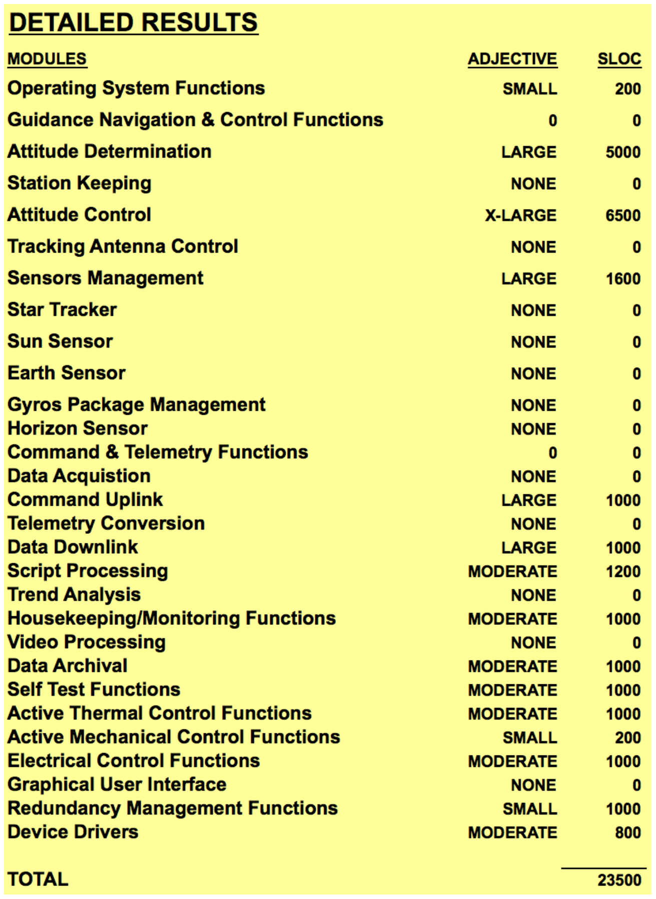
III. Software Development Process

Key to the software estimation effort was the implementation of the agile model-based software development process shown in Figure 2. For LADEE, the model based development effort using the Workstation Simulation (WSIM) environment allowed early verification that the algorithms and software design met the detailed control requirements. Software would then be generated in the "C" programming language, and then integrated with the other software layers and tested in real time Processor-in-the-Loop (PIL) and Hardware-in-the-Loop (HIL) environments. One HIL test-bed was denoted the "Travelling Road Show", and could be taken to payload development sites so that early integration testing could be performed. Once the data was generated from all of the various test-beds, it would be integrated into an automated reporting system that provided bidirectional traceability between requirements, models or code and the test results. The report system also automatically generated a requirements verification matrix that was used to focus the development effort. Verification of each level 4 requirement was assigned to 4 of the 5 major build development cycles as part of the concept of operations and software build specifications. The final build cycle was reserved for defect reduction for items found in spacecraft Integration and Test (I&T).

IV. Software Size Estimation

A key component of parametric cost estimation is an estimate of Source Lines of Code (SLOC). LADEE utilized the NASA Marshall (MSFC) "Source Lines of Code Estimator" spreadsheet, which quantifies source lines of code to be developed given adjective descriptions of the size and complexity of the unit. An image of the detailed results from the spreadsheet is provided in Table 2. Comparing the software architecture in Figure 1 to the categories in the SLOC spreadsheet reveals that the LADEE architecture did not map one-to-one with the general categories in the MSFC spreadsheet. For instance, The Safe Mode Controller (SMC), and Attitude Control System (ACS) both map into "Attitude Control" and "Gyro Package Management", but "Sun Sensing" is part of the SMC system alone. The State Estimation Module (EST) has aspects of "Star Tracker", "Attitude Determination". Other systems map more closely, such as Command Ingest (CI) and Telemetry Output (TO) with "Command Uplink" and "Data Downlink". It can be seen that the Memory Scrub software was not included in the MSFC worksheet estimate at all because it was not identified as a requirement at PDR.

Table 1. SLOC estimate from MSFC estimation spreadsheet.



DETAILED RESULTS

MODULES	ADJECTIVE	SLOC
Operating System Functions	SMALL	200
Guidance Navigation & Control Functions	0	0
Attitude Determination	LARGE	5000
Station Keeping	NONE	0
Attitude Control	X-LARGE	6500
Tracking Antenna Control	NONE	0
Sensors Management	LARGE	1600
Star Tracker	NONE	0
Sun Sensor	NONE	0
Earth Sensor	NONE	0
Gyros Package Management	NONE	0
Horizon Sensor	NONE	0
Command & Telemetry Functions	0	0
Data Acquistion	NONE	0
Command Uplink	LARGE	1000
Telemetry Conversion	NONE	0
Data Downlink	LARGE	1000
Script Processing	MODERATE	1200
Trend Analysis	NONE	0
Housekeeping/Monitoring Functions	MODERATE	1000
Video Processing	NONE	0
Data Archival	MODERATE	1000
Self Test Functions	MODERATE	1000
Active Thermal Control Functions	MODERATE	1000
Active Mechanical Control Functions	SMALL	200
Electrical Control Functions	MODERATE	1000
Graphical User Interface	NONE	0
Redundancy Management Functions	SMALL	1000
Device Drivers	MODERATE	800
TOTAL		23500

While the mismatch in categories causes some difficulty for the analysis, a comparison between the actual software SLOC and the MSFC estimates can be performed for some units. Table 2 provides the software SLOC for the final flight software version for LADEE. The SLOC count has been computed using David Wheeler's "sloccount" [9]. Care had to be taken to not include auto-generated test functions for the Simulink module. The MSFC spreadsheet predicted approximately 1000 lines of code for each of the hand generated CI and TO modules, which represented a small overestimate for CI (807) and a moderate underestimate for TO (1405). This may be because the estimated complexity for the TO routine was too low, and indeed increasing the complexity to "X-large" would have increased the SLOC estimate to 1500, very close to the final outcome. The basis for the SLOC count in the MSFC spreadsheet is also unknown, and a mismatch in the methodology could have an impact on these comparisons. Looking at a representative auto-generated unit, TCS was reasonably estimated at 1000 vs. 1139 actual SLOC. This unit was largely composed of embedded Matlab components, so is an interesting case to look at the size of the auto-code vs. the hand-coded part of the unit. In particular, the embedded Matlab scripts for the unit had only 120 SLOC, while the main Simulink function contained 680 SLOC, while the normally generated wrappers comprised the rest of the code. So, in these cases, even for auto-code, the estimates were quite good.

The main GN&C control functions were significantly underestimated for the flight software. Attitude determination was estimated as 5000 SLOC, while the actual was 8372. Had we categorized the function as "X-Large", the estimate would have been 7500, and in retrospect, this would have been a better choice, but at PDR we assumed the operation of the star tracker would be straight-forward, when indeed, it became quite complex. The SMC and ACS modules would both be included in the estimate of attitude control functions with actual SLOC of 8628 well in excess of the "X-Large" estimate of 6500 at PDR. The biggest discrepancy in the SLOC estimate was the underestimation of the hardware I/O drivers and the memory scrub module needed for the mission. They comprised about 12500 SLOC, which is a significant portion of the difference between the estimate of 23500 shown in Table 3 and actual newly developed of 41430 SLOC. The newly developed SLOC even greatly exceeded our estimated high SLOC of 29500. To generate the high and low estimates, we tweaked the size of each unit one up and one down from the most likely values we chose.

V. COCOMO Based Effort Estimation

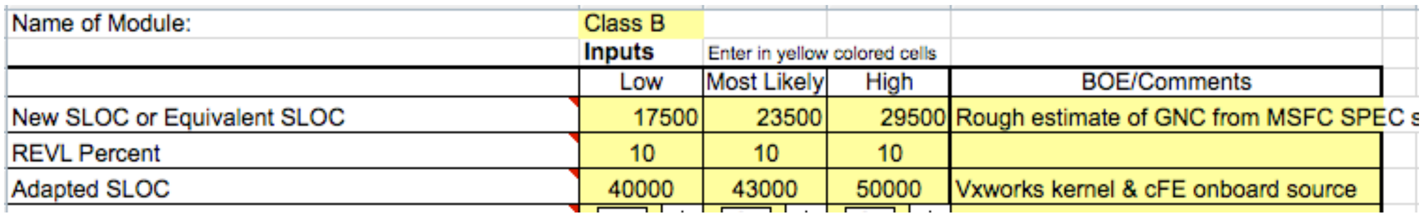
The SLOC estimates developed in the previous section were used as part of a COCOMO-based [10] effort estimation using the Jet Propulsion Laboratory (JPL) Software Cost Analysis Tool (SCAT) spreadsheet [11]. This spreadsheet utilizes the SLOC estimates along with other parameters such as software complexity, skill of the team, percentage software reuse, and many other parameters. The effort multipliers and scale factors are set qualitatively from low to high with descriptive language accompanying each factor, useful for a novice trying to tune parameters appropriately in the sheet. Once the parameters are tuned, one can then generate Monte-Carlo results for the total effort and cost likely needed for the mission. As seen in Figure 3, at PDR, the 70% estimate for the effort to develop the Class B software was listed as approximately 28 Full Time Equivalents (FTE) from development of the Class B software requirements through the software I&T period. This did not include support for mission operations

As a thought experiment on impact of software reuse, we modified the numbers in Table 3 such that the adapted SLOC line was reduced by 40K and the new SLOC line was increased by 40K, leaving all other factors the same. This somewhat simulates the situation of re-developing the cFE/cFS/OSAL services from our estimate of the software size at PDR. Surprisingly, the predicted work effort only increased to approximately 29.5 FTE. This is counter-intuitive, as we do not believe that we could achieve the level of services and quality by redeveloping the executive services layer rather than reusing it. It more than likely reflects the

Table 2. Software SLOC for final flight software load.

CSU	Description	Category	SLOC
TCS	Thermal Control System	Auto-generated, New	1139
PCS	Power Control System	Auto-generated, New	2201
BCS	Battery Control System	Auto-generated, New	2139
CMP	Command Mode Processor	Auto-generated, New	1655
ACT	Actuator Manager	Auto-generated, New	2498
ACS	Attitude Control System	Auto-generated, New	4394
EST	State Estimator	Auto-generated, New	8372
SMC	Safe Mode Controller	Auto-generated, New	4234
CI	Command Ingest	Hand Generated, New	807
TO	Telemetry Output	Hand Generated, New	1405
MS	Memory Scrub	Hand Generated, New	2785
HWIO	Collection of Hardware I/O Drivers	Hand Generated, New	9801
GOTS	cFE/cFS/OSAL components	Some Adaptation.	66771
OS	VxWorks/Drivers	OS: Full Reuse, Drivers Modified	37694
New Development			41430
Total			145895

Table 3 Estimates of New, Modified and Adapted SLOC at PDR



Name of Module:	Class B			
	Inputs	Enter in yellow colored cells		
	Low	Most Likely	High	BOE/Comments
New SLOC or Equivalent SLOC	17500	23500	29500	Rough estimate of GNC from MSFC SPEC s
REVL Percent	10	10	10	
Adapted SLOC	40000	43000	50000	Vxworks kernel & cFE onboard source

relatively high cost estimate of adapting and testing re-used software.

Another interesting experiment is to take the final flight software SLOC counts from Table 2 with all other factors remaining the same in the SCAT sheet and redo the Monte-Carlo calculation. In this case, the 70% level of effort is estimated as 33 FTE. A final expement is to evaluate the spreadsheet for the case where the entire software SLOC count is listed as new with no reuse at all. In this case, the work effort is estimated at 70 FTE.

VI. Bottoms up Level of Effort

In this approach, we developed a detailed Work Breakdown Structure (WBS) for the project based on our experience with the hover test vehicle rapid prototyping process. For each Computer Software Component (CSC), we estimated the number Computer Software Units (CSU) that would be required to implement the functionality needed. Based on our prototyping effort to develop the hover test vehicle, we estimated that it would take 10 hours to implement each basic unit in either Simulink or hand code. The 10 hours did not include the development of the requirements, design, integration, test or base-lining of the unit. We then took a poll across the project management soliciting estimates of multiplies applied to the base development to perform the rest of the required software activities. Estimates for the multiplier ranged from a low of 5 to a high of 20 hours per hour of actual software development. These were then calibrated against the hover test effort to settle on a multiplier of 7.7

Project management activities were estimated separately from the sheer software development effort. For project management, including documentation, quality system setup and top level system engineering, we estimated the hours of effort by each discipline lead for each Work Breakdown Structure (WBS) element on a per build, per document or per week as necessary. For instance, we estimated 16-24 hours per discipline lead per major review (such as the Preliminary Design Review). This turned out to be a significant underestimate as the effort usually turned out to be more than a week per person. Another major effort that was not captured in our estimates was the hundreds of hours required for certification and recertification in Capability Maturity Model (CMMI) [12] Level 2 over the course of the mission. The program management related estimates were integrated with the software effort estimates and added up in a master spreadsheet. Through this process, we estimated the total software development effort from the "Bottoms Up" estimate to require approximately 27 FTEs.

VII. IMDC EFFFORT

The Flight Software team also participated in an early costing exercise for the overall LADEE mission that was conducted by the Integrated Mission Design Center (IMDC) at NASA Goddard Spaceflight Center. The IMDC performs rapid concept studies for future missions using a skilled resident engineering team working closely with the customer team in a collaborative, concurrent design environment. As part of the costing exercise, IMDC personnel examined the types of functionality that needed to be provided for the mission by the Flight Software. The team elicited mission and spacecraft parameters, such as numbers of sensors and actuators, software reuse, CPU type and capabilities, test-bed requirements, etc.. The spreadsheet is calibrated based on previous experience for the level of effort for each of these areas of functionality. The spreadsheet rollup breaks out the FTE effort into 6 functional areas and includes procurement estimates. The LADEE mission was estimated to take 38.5 FTEs

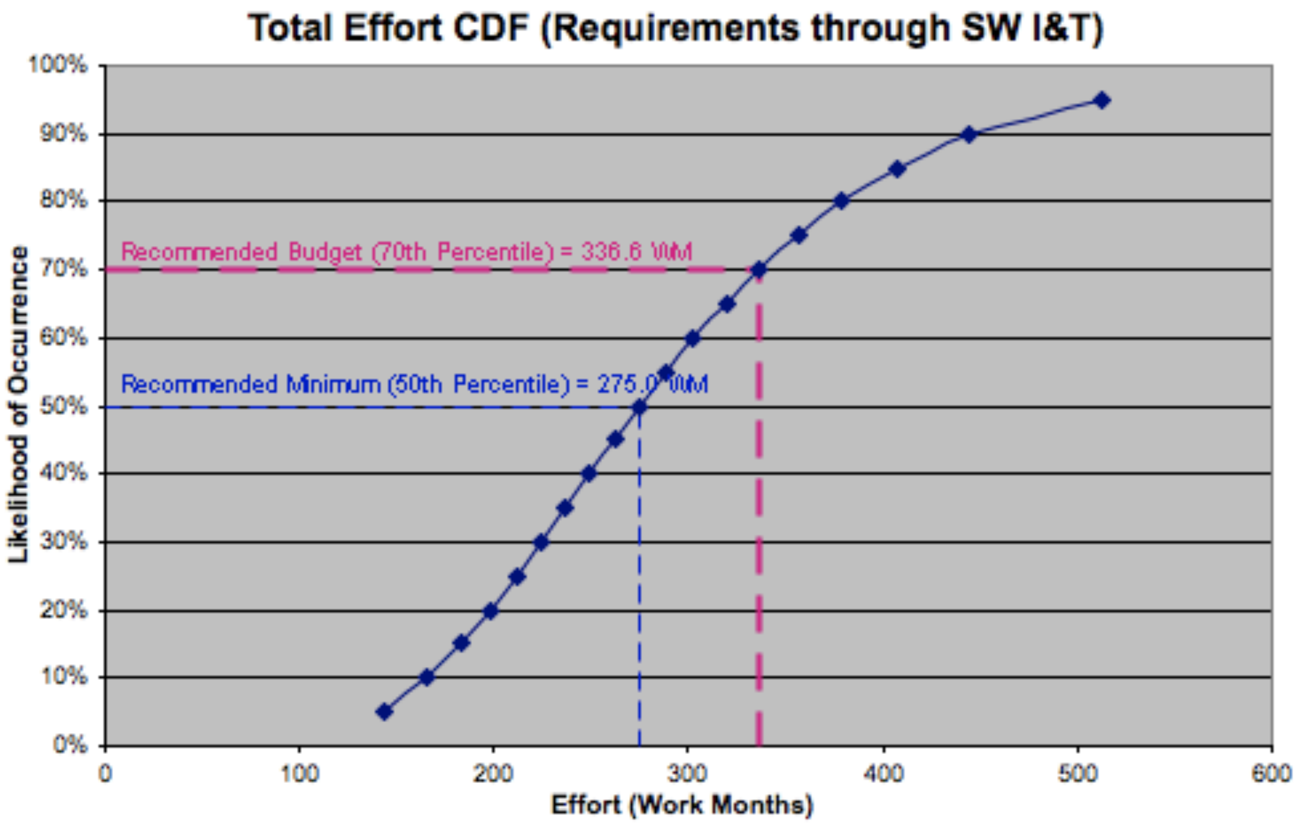


Figure 3. Estimation of Effort from JPL SCAT spreadsheet

and $1.1M in procurement from start of development through pre-launch preparation. Post launch estimates are rolled up into Mission Operations budgets. Details of the IMDC methodology can be found in [13].

VIII. Results and Discussion

Table 4 shows the labor and overall cost that was estimated using the three methodologies previously described. As the table shows, the estimates were relatively similar, and closely match the actual costs that were recorded. One difference between the methods is that support from GSFC was counted as procurement dollars rather than FTE in the Bottoms up and parametric estimations. It is apparent though, that the vast majority of costs for flight software development are from personnel time rather than raw procurement. The IMDC methodology was closest, remarkably within 2% of the actual cost of the flight software effort. The higher level of accuracy for the IMDC method is somewhat expected because it was built on years of experience in cost estimation. Given perfect knowledge at PDR of the ultimate SLOC count of the software, the COCOMO parametric estimation method would have been remarkably close to the final actual effort (33 vs. 33.7 FTE). Even with the various mistakes, all three methods were relatively close and well within 10% of the ultimate cost of the software. This adds confidence that the early cost estimates were at least reasonable despite the inexperience of the team.

It should be noted that in the early stages of the project, the Flight Software and Project Management teams chose to utilize the lowest of the three estimates as an optimistic plan, while keeping in sufficient reserve to meet the cost of the highest estimate should the cost of the Flight Software exceed that amount.

While the estimates are all quite close to the actuals, it should be noted that the flight software team for LADEE was very invested in the software and well-being of the spacecraft, and worked many hours that are not reflected in the official totals. The internal software cost estimations techniques also generally under predicted the totals, in part, because we responded to management concerns about the size of the software budget. We also did not properly estimate the extraordinary amount of time used for official reviews, key decision points, and for the CMMI certification process that occurred multiple times during the development process.

IX. Conclusions

The software cost estimation process for the LADEE program produced good estimates despite the relative inexperience of the members involved. The use of the MSFC SLOC estimation spreadsheet could have been improved by making the software structure more closely aligned with the structure of the spreadsheet, but it is not appropriate to impose structure on the software architecture due to the cost estimation process. The COCOMO parametric analysis is remarkably good given a decent estimate of the code size even for a model-based development effort. This is surprising, because the effort of developing functionality in a graphical modeling environment is only loosely related to actual SLOC count. The auto-coding template and methodology used can impact SLOC counts. There are many rumors of "software bloat" due to model-based techniques that were not born out in this development effort, perhaps due to the templates used. In the end, the software was developed on schedule and relatively close to all of the estimates, with the IMDC clearly being the closest to actuals. Once calibrated for the particular type of development methodology used, parametric cost estimation such as COCOMO can be quick and effective. For an inexperienced team on a new-start project, it is recommended that multiple estimations be used to improve confidence in the proposed budget and schedule.

Table 4 Estimates versus Actual Costs

	Estimated FTE	Estimated Procurements	Overall Cost*
Parametric Model	28	$2.5 M	$9.5 M
Bottoms Up Analysis	27	$2.5 M	$9.3 M
IMDC Estimate	38.5	$1.1 M	$10.7 M
Actuals	33.7	$2.49 M	$10.9 M

*Average cost for FTE assumed at $250K/yr.

REFERENCES

[1] Delory, G.T.; Elphic, R.; Morgan, T.; Colaprete, T., Horanyi, M;, Mahaffy, P.; Hine, B.; Boroson, D, "The Lunar Atmosphere and Dust Environment Explorer (LADEE)", 40th Lunar and Planetary Science Conference, (Lunar and Planetary Science XL), held March 23-27, 2009 in The Woodlands, Texas, id.2025

[2] Hine, B., Turner, M., Marshall, W. S. , "Prototype Common Bus Spacecraft: Hover Test Implementation and Results", NASA/TM-2009-214597.

[3] Trivailo O, et al. Review of hardware cost estimation methods, models and tools applied to early phases of space mission planning. Progress in Aerospace Sciences (2012)

[4] Mathworks Corporation, http://www.mathworks.com.

[5] Wildermann, Charles P, "NASA/GSFC's Flight Software Core Flight System", Flight Software Workshop, http://flightsoftware.jhuapl.edu/, 2008.

[6] McComas, David, "NASA/GSFC's Flight Software Core Flight System", Flight Software Workshop, http://flightsoftware.jhuapl.edu/, 2012.

[7] Wind River Corporation http://www.windriver.com/products/vxworks/

[8] http://standards.ieee.org/develop/wg/POSIX.html

[9] Wheeler, David. http://www.dwheeler.com/sloccount/

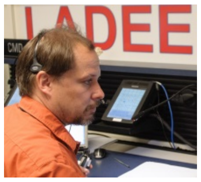
[10] COCOMO website http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

[11] Lum, Karen. "Software Cost Analysis Tool User Document", Caltech/NASA Jet Propulsion Laboratory, 2005.

[12] Paulk, Mark C.; Weber, Charles V; Curtis, Bill; Chrissis, Mary Beth (1995). The Capability Maturity Model: Guidelines for Improving the Software Process. SEI series in software engineering. Reading, Mass.: Addison-Wesley. ISBN 0-201-54664-7.

[13] Karpati, G.; Martin, J.; Steiner, M.; Reinhardt, K., "The Integrated Mission Design Center (IMDC) at NASA Goddard Space Flight Center," Aerospace Conference, 2003. Proceedings. 2003 IEEE , vol.8, no., pp.8_3657,8_3667, March 8-15, 2003

Biography



Howard Cannon *is a Computer Engineer at NASA Ames Research Center. He served as the LADEE Flight Software Subsystem Lead throughout the formulation phase and most of the spacecraft development phase. He then served as a Flight Director in Mission Operations for the remainder of the mission. As the Flight Software Lead, he participated in the development of the estimates in order to negotiate and establish the software budget. He is currently leading the software development effort for the Resource Prospector mission. Cannon has a B.S. in Mechanical Engineering from Bradley University in Peoria IL, and an M.S. in Robotics from Carnegie Mellon University.*

Karen Gundy-Burlet *is a Research Scientist at NASA Ames Research Center. Se served as the LADEE Flight Software Subsystem Quality Engineer, became the Flight Software Lead and served in Mission Operations. She led the development of the LADEE FSW V&V system and performed other tasks pertinent to this report such as software cost estimation. She currently leads the Common Avionics and Software Technologies (CAST) team that is generalizing the LADEE FSW to other spacecraft missions. Dr. Gundy-Burlet graduated with a B.S. in Mechanical Engineering from U.C. Berkeley and obtained M.S. and Ph.D. degrees in Aerospace Engineering from Stanford.*