

# The NASA Analogy Software Cost Model: A Web-Based Cost Analysis Tool

Jairus Hihn, Michael Saing,  
Elinor Huntington  
Jet Propulsion Laboratory,  
California Institute of Technology  
Pasadena, CA 91109

James Johnson  
National Aeronautics and Space  
Administration  
Washington, DC

Tim Menzies, George Mathew  
North Carolina State University  
Raleigh, NC

*Abstract*—This paper provides an overview of the many new features and algorithm updates in the release of the NASA Analogy Software Cost Tool (ASCoT). ASCoT is a web-based tool that provides a suite of estimation tools to support early lifecycle NASA Flight Software analysis. ASCoT employs advanced statistical methods such as Cluster Analysis to provide an analogy based estimate of software delivered lines of code and development effort, a regression based Cost Estimating Relationships (CER) model that estimates cost (dollars), and a COCOMO II based estimate. The ASCoT algorithms are designed to primarily work with system level inputs such as mission type (earth orbiter vs. planetary vs. rover), the number of instruments, and total mission cost. This allows the user to supply a minimal number of mission-level parameters which are better understood early in the life-cycle, rather than a large number of complex inputs.

## TABLE OF CONTENTS

1. INTRODUCTION .....	1
2. DATA SUMMARY .....	2
A. Sources.....	2
B. Data Description .....	2
3. METHODOLOGY.....	5
4. RESULTS .....	5
A. Analogy Cluster Model Results.....	5
B. Regression Results .....	6
5. NEXT STEPS.....	11
REFERENCES.....	11
BIOGRAPHY.....	11

## 1. INTRODUCTION

The prototype version of ASCoT, the NASA Analogy Software Cost Tool, was first introduced at the 2016 IEEE Aerospace conference [1]. ASCoT is designed to address the specific problems associated with the issues of sparse, small, and noisy data sets. The NASA Analogy Software Cost Model is built on research into the effectiveness of data mining algorithms over the past ten years by Menzies et al [2,3,4,5]. The model uses a combination of spectral clustering and k-nearest neighbor on system characteristics, which are symbolic and not numerical data. This enables the ability to estimate software development effort early in the project lifecycle with easily attainable inputs like the type of mission and the number of instruments. ASCoT is

developed as a compliment or extension to the existing widely used parametric methods. The other contribution of this paper is the emphasis on the use of the magnitude of relative error (MRE) and it's associated measures of Median MRE (MMRE) and Pred(30)<sup>1</sup> as a metric for evaluating cost model performance across very different types of models.

The following is a summary of the key findings from our previous work [1]:

- There are a variety of models whose performances is hard to distinguish (given currently available data), but some models are clearly better than others.
- If one has sufficient detailed data to run COCOMO or a comparable parametric model, then the best model is the parametric model. On this point also see [6].
- When insufficient information exists then a model using system parameters only can be used to estimate software costs with only a small reduction in accuracy. The main weakness is the possibility of occasional large estimation errors which the parametric model does not exhibit.
- While a nearest neighbor model performs as well as spectral clustering based on MMRE, spectral clustering handles outliers better and provides a structured model with more capability.
- A major strength of the nearest neighbor and spectral clustering methods is the ability to work with a combination of symbolic and numerical data.

ASCoT was initially developed with a Microsoft Excel™ based front end and a Python coded back end, which read from an Excel input file. The new release of ASCoT is a totally web-based application with an associated database that is only available to the NASA community by NASA Headquarters (HQ) approved users at <https://www.ONCEData.com>. This demonstrates NASA's capability of moving advanced statistical models to the online environment, which we hope to build on in the

<sup>1</sup> Pred(30) was a popular measure of model performance in the eighties and nineties, but seems to have fallen out of favor.

future. The implementation of a web-based model has allowed ASCoT to expand its capabilities more quickly. For example, the use of data visualization has been greatly increased and ASCoT can now run on both PCs and Macs.

There are also number of other changes that have been made to ASCoT over the past year. There is the addition of new historical data, which has enabled us to refine the clusters and improve the CER. The parametric model, COCOMO II has also been added. COCOMO II uses the outputs from the ASCoT spectral clustering algorithms to derive the more detailed inputs required by COCOMO. The biggest change is – saying, “Good Bye!” to Microsoft Excel™.

## 2. DATA SUMMARY

### A. Sources

The NASA Cost Analysis Data Requirement (CADRe) is a formal project document that describes the life-cycle cost, schedule, technical, and risk information of a project. The CADRe has three separate Parts: A, B, and C. Part A is a narrative description of the project throughout its lifecycle at each milestone and includes essential subsystem descriptions, block diagrams, and heritage assumptions. Part B contains the technical design parameters such as power, mass, and software metrics for each subsystem in a standardized template. Part C captures all the cost data broken out by Work Breakdown Structure (WBS) throughout the lifecycle by project phase.<sup>2</sup>

- ASCoT Data was updated last in October, 2016
- Available missing data items were obtained from other sources including contacting project software managers
- Verifiable CADRe data was revised with information/data from other sources
- System descriptor data was supplemented with data from NASA project websites, project reports, and Wikipedia articles.
- Software metrics for older missions that predated the CADRe were supplemented with data records from a data collection conducted for the International Space Station that was completed in 1990. A subset of these records can be found at the PROMISE (Predictor Models in Software Engineering) website under the COCOMO directory.
- Contributed Center level data

The data analysis used for Table 1 through Table 14 varies in “Number (#s) of Records” due to some incomplete data which was not used. The analysis only accounts for complete data, for that particular analysis, as indicated for each table result.

### B. Data Description

Table 1 contains a list of the data used in the study including the total number of mission records that have data by each variable. For a detailed description of the types of data parameters collected see Appendices A (COCOMO Model Inputs) and B (System Parameters). There is a total of 61 missions where data was collected, but a few were not used

due to partially incomplete data. The number of projects for which there was at least partial data increased by 10 to 49 projects. Of those only 34 could be used to update the cluster model and 37 for the regression model. There is also an increase in the correctness of the data from the prototype release.

Table 1 Data summary with number of records – 34 missions have complete verified data and were used in ASCoT Clusters

Data Item	Number of Missions (Current - 2017)	Number of Missions (2016)
Total development effort in work months	36	28
Flight Software Development Cost	37	30
Flight System Development Cost	37	30
<b>Logical Lines of code (LOC)</b>		
Delivered LOC	49	36
Inherited LOC (Reused plus Modified reused lines)	43	36
COCOMO model inputs (See Appendix A for the parameter definitions) - Translated from CADRe which has SEER model inputs because the SEER data items are very sparse in CADRe	19	19
<b>System parameters * (See Appendix B parameter definitions)</b>		
Mission Type (deep-space, earth-moon, rover-lander, observatory)	49	39
Multiple element (probe, etc.)	49	39
Number of Instruments	49	39
Number of Deployables	49	39
Flight Computer Redundancy (Dual Warm, Dual Cold, Single String)	49	39
Software Reuse (Low, Medium, High)	41	36
Software Size (Small, Medium, Large, Very Large)	41	36

Table 2 contains a list of all missions for which data was obtained with an indication of which missions were used to build the analogy and regression models. While the two models share many of the same projects the data used is different as ASCoT is an effort estimation model and the regression is a cost estimation model with software cost as a function of spacecraft cost.

<sup>2</sup> [https://www.nasa.gov/sites/default/files/files/CEH\\_AppA.pdf](https://www.nasa.gov/sites/default/files/files/CEH_AppA.pdf)

Table 2 Mission by type and model inclusion

Mission Type	Mission	ASCoT	Regression
Deep Space	Cassini	X	
Deep Space	Contour		X
Deep Space	Dawn	X	X
Deep Space	Deep Impact	X	X
Deep Space	DS1	X	X
Deep Space	Genesis	X	X
Deep Space	GLL	X	
Deep Space	JUNO	X	X
Deep Space	LADEE		X
Deep Space	MAP		X
Deep Space	Mars Odyssey	X	
Deep Space	Maven	X	
Deep Space	Messenger	X	X
Deep Space	MRO	X	X
Deep Space	NEAR	X	X
Deep Space	New Horizons	X	X
Deep Space	OSIRIS REX	X	X
Deep Space	Stardust	X	
Deep Space	Van Allen Probe (RBSP)	X	X
Observatory	GRO	X	
Observatory	HST	X	
Observatory	Kepler	X	
Observatory	Stereo	X	X
Observatory	WISE		X
Earth/Lunar Orbiter	AIM		
Earth/Lunar Orbiter	Aqua		
Earth/Lunar Orbiter	EO1		
Earth/Lunar Orbiter	FAST		X
Earth/Lunar Orbiter	GALEX		
Earth/Lunar Orbiter	GEMS	X	
Earth/Lunar Orbiter	GEOTAIL		
Earth/Lunar Orbiter	GLAST		
Earth/Lunar Orbiter	GLORY	X	X
Earth/Lunar Orbiter	GOES R	X	
Earth/Lunar Orbiter	GPM Core	X	X
Earth/Lunar Orbiter	Grail	X	X
Earth/Lunar Orbiter	IBEX		X
Earth/Lunar Orbiter	IRIS		
Earth/Lunar Orbiter	LCROSS		
Earth/Lunar Orbiter	LDCM		
Earth/Lunar Orbiter	LRO	X	X
Earth/Lunar Orbiter	NOAA-N-Prime		
Earth/Lunar Orbiter	NPP		
Earth/Lunar Orbiter	NuStar	X	X
Earth/Lunar Orbiter	OCO	X	X
Earth/Lunar Orbiter	OCO 2		X
Earth/Lunar Orbiter	OCO 3		
Earth/Lunar Orbiter	RHESSI		
Earth/Lunar Orbiter	SAMPEX		X
Earth/Lunar Orbiter	SDO	X	X
Earth/Lunar Orbiter	SMAP	X	
Earth/Lunar Orbiter	SWAS		X
Earth/Lunar Orbiter	TIMED	X	X
Earth/Lunar Orbiter	TRACE		X
Earth/Lunar Orbiter	TRMM		X
Earth/Lunar Orbiter	WIRE		X
Rover	MER	X	X
Rover	MPF	X	X
Rover	MSL	X	X
Static Lander	Insight	X	X
Static Lander	Phoenix	X	X

Tables 3 through 8 below summarize the data by median, average, and spread metrics for each parameter. There was little change in the summary metrics as a result of the addition of the new and corrected data. Overall Deep Space missions have more lines of code, higher development effort, cost more, have more instruments, and are more likely to be dual string than Earth Orbiters. Not

surprisingly, In Situ missions have significantly more deployables and instruments than all other mission types. Slightly surprising is that Earth Orbiters and deep space missions have similar inheritance rates even though many Earth orbiters can draw more easily on the various contractor product lines.

Table 3 Effort by mission type

Mission Type	EFFORT (months)				
	# Records	Median	S.D.	Avg.	Range
Earth/Lunar Orbiter	22	584	354	651	100 – 1,190
Observatory	5	492	631	742	233 – 1,830
Deep Space	17	637	375	686	48 – 1,436
In Situ	5	1,080	555	1,232	634 – 1,888

Table 4 Delivered LOC by mission type, actual count

Mission Type	Logical Delivered LOC				
	#Rec.	Median	S.D.	Avg.	Range
Earth/Lunar Orbiter	22	96,000	41,432	101,821	12,000 – 170,000
Observatory	5	107,000	95,548	23,000	23,000 – 280,000
Deep Space	17	122,000	75,431	24,000	24,000 – 289,900
In Situ	5	205,000	145,334	94,3000	94,300 – 475,000

Tables 5 and 6 show software size and inheritance by categories. While actual code counts or estimated percent existed for software size and inheritance, these values were converted to categories for two reasons. Most notably, the model under development is designed to be used in early lifecycle phases and estimators would only have an approximate idea as to the number of delivered and inherited LOC. The other reason is that there are many inconsistencies in how lines of code are recorded, which impacts the NASA CADRe, and the counting rules used are often not documented, so the use of categories is a more accurate reflection of the actual accuracy of the data.

Table 5 Software size by size category and mission type

Mission Type	Software Size					
	#Rec.	Very Low to None	Low	Med	High	Very High
Earth/Lunar Orbiter	22	3	13	6	0	Medium
Observatory	6	1	5	0	0	Medium
Deep Space	16	2	4	7	3	Large
In Situ	5	0	1	2	2	Large

Table 6 Inheritance by Mission Type

Mission Type	Inheritance						
	#Rec.	Very Low to None	Low	Med	High	Very High	Med.
Earth/Lunar Orbiter	18	4	0	4	4	6	High
Observatory	5	0	1	2	1	1	Low
Deep Space	15	2	3	2	3	5	High
In Situ	5	2	1	0	1	1	Very Low/None

Table 7 and 8 show deployables, instruments, and flight computer redundancy by mission types. For deployables and number of instruments, it is shown that these numbers are high for Deep Space and In Situ compared to Earth/Lunar Orbiter and Observatory. The flight computer redundancy's Dual-String Cold and Dual String Warm is also shown to be higher compared to Earth/Lunar Orbiter and Observatory missions.

Table 7 Deployables and instruments by mission type

Mission Type	#Rec.	Deployables		Instruments	
		Median	RANGE	Median	RANGE
Earth/Lunar Orbiter	22	2	0 - 7	3	1 - 7
Observatory	6	2	1 - 6	4	1 - 6
Deep Space	16	2	0 - 8	5	2 - 12
In Situ	5	6	2 - 10	7	3 - 10

Table 8 Flight computer redundancy by mission type

Mission Type	Flight Computer Redundancy				
	#Rec.	Single String	Dual-String Cold	Dual-String Warm	Median
Earth/Lunar Orbiter	22	14	8	0	Single String
Observatory	6	1	5	0	Dual String Cold
Deep Space	16	1	13	2	Dual String Cold
In Situ	5	1	0	4	Dual String Warm

Tables 9 through 12 shows Delivered Productivity by Logical Lines of Code by mission type and inheritance level; low (<20%), medium (<50%), high to very high (>=50%). Inherited code includes both reused and modified reused code reuse. As expected, all mission categories clearly show that increases in inheritance result in higher productivity rates.

Table 9 Productivity (Delivered Logical LOC) by mission type

Mission Type	Productivity (Logical Del/month)				
	# Records	Median	S.D.	Avg.	Range
Earth/Lunar Orbiter	22	191	214	260	65 - 823
Observatory	5	244	192	238	46 - 460
Deep Space	17	208	168	262	37 - 615
In Situ	5	249	81	212	87 - 292

Table 10 Very low to none and low inheritance delivered productivity

Very Low to None and Low Inheritance (0% - <20%) DELIVERED Productivity				
Mission Type	# of Records	Avg. Prod	Median Prod	Range
Earth/Lunar Orbiter	4	106	106	62 - 150
Observatory	1	-	-	-
Deep Space	5	134	130	24 - 214
In Situ	3	292	308	94 - 475

Table 11 Medium inheritance delivered productivity

Mission Type	Medium Inheritance (>=20% - <50%) Delivered Productivity			
	#Records	Median	Avg.	Range
Earth/Lunar Orbiter	4	96	12 - 170	94
Observatory	2	66	23 - 109	66
Deep Space	2	141	100 - 182	141
In Situ*	-	-	-	-

Table 12 High and very inheritance delivered productivity

High and Very High Inheritance (>=50%) Delivered Productivity				
Mission Type	# of Records	Avg. Prod	Median Prod	Range
Earth/Lunar Orbiter	10	99	95	41 - 156
Observatory	2	194	194	107 - 280
Deep Space	8	169	146	86 - 290
In Situ	2	195	195	185 - 205

Tables 13 and 14 provide a summary of the flight software and flight system cost data in FY16 dollars (\$K). As with the effort data, the cost of deep space missions are more

expensive than earth orbiters. The data indicates that the difference in cost is greater than the difference in effort between mission types. This is most likely because the reported cost includes procurements and costs of additional WBS elements that were not included in the effort data. One example is simulators for the flight system which some contractors include with flight software costs as they are used for testing the flight software. Another pattern not shown here, but is in the data, is that the median value of the ratio of flight software costs to flight system cost are 10% for all mission types except In Situ which is 5%.

Table 13 Software development cost (FY16\$ K)

Mission Type	Software Development Cost (FY16\$ in K)				
	# Rec.	Median	S.D.	Avg.	Range
Earth/Lunar Orbiter	17	\$6,653	\$8,542	\$10,141	\$1,134 - \$24,205
Observatory	3	\$8,506	\$7,093	\$11,559	\$6,504 - \$19,667
Deep Space	13	\$14,445	\$9,836	\$15,311	\$1,102 - \$39,951
In Situ	4	\$22,650	\$24,697	\$28,881	\$7,286 - \$62,940

Table 14 Total spacecraft (FY16\$ K)

Mission Type	Total Spacecraft Cost (FY16\$ in K)				
	# Rec.	Median	S.D.	Avg.	Range
Earth/Lunar Orbiter	17	\$61,498	\$63,247	\$85,998	\$14,798 - \$200,398
Observatory	3	\$62,822	\$100,212	\$114,039	\$49,786 - \$229,508
Deep Space	13	\$169,302	\$94,343	\$197,008	\$84,971 - 401,063
In Situ	4	\$381,995	\$538,288	\$570,111	\$181,375 - 1,335,078

### 3. METHODOLOGY

One of the significant contributions of the research conducted in developing ASCoT is the recognition of the importance of the use of the magnitude of relative error (MRE) and its associated measures such as the mean or median MRE as a metric for evaluating cost model performance across very different types of models. MRE measures are popular in the data mining literature because they require no assumptions about the underlying distributions enabling one to compare model performance across very different types of models. Pred(30), another MRE statistic, was very popular in the cost field in the eighties and nineties but seems to have fallen out of favor [7]. Our research indicates one should use the entire MRE curve, the Median MRE (MMRE), the interquartile range, and then when appropriate compute the Pred measure based on the risk perception of different types of error. For example, an evaluation metric for models of Pred(30)>80% is a metric that says, “be accurate most of the time”, and it is acceptable if the model is way off, “once in a while.” A Median MRE (MMRE) metric indicates that one is

concerned with model performance across the *entire range* of the data. For a detailed description of the models evaluated and the evaluation method see [1, 4]. Hihn et al (1) also contains a detailed description of the spectral clustering algorithm used in ASCoT.

The regression based cost model included in ASCoT was derived using standard linear regression based on the F-test results of the F-test, t-test and R2. The cost regression is derived from a related but different set of data than the analogy model using the cost data in CADRe Part C and not the development effort as reported in CADRe Part B.

The only difference in the methodology from that reported is [1] how the test cases were generated. In [1] we used leave out validation while in this paper we generated a set of 12 test cases derived from a standard decomposition of the data set based on mission type, software size, software inheritance and then took the average or median of records in each subset as appropriate.

### 4. RESULTS

#### A. Analogy Cluster Model Results

The cluster based analogy model of ASCoT Beta is a significant improvement over the previous versions of the model. The median MRE decreased by a third from the prototype model as can be seen in Table 15 and in Figure 1. The improvement is a result of the increased sample size and data corrections as more and better data has become available from the NASA CADRe’s. There is especially a large improvement in the reduction in large errors with the worst case decreasing from 506% to 175% and now 11 of 12 or 92% of the test cases have a relative error of less than 55%. However neither of the models yet meets the “old gold standard” of Pred(30) ≥ 80%. It is expected that the results will continue to improve as more data becomes available.

Table 15 MRE by rank order and model version

MRE by Rank Order and Model Version		
	ASCoT Beta	ASCoT Prototype
	1%	0%
	3%	1%
	3%	3%
	10%	4%
	22%	4%
	23%	35%
	29%	45%
	35%	79%
	37%	101%
	51%	102%
	54%	192%
	175%	506%
Median MRE	26%	40%
Mean MRE	37%	89%

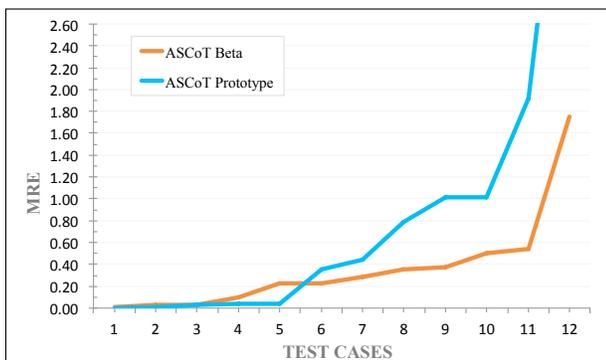


Figure 1 MRE by rank order and Model Version

The updated analogy model has 9 clusters compared to 8 and many of the clusters now have more than 3 projects in each cluster making them more robust. The current cluster set is summarized in Table 16 which shows the cluster membership and the median development effort.

An overview of how the clusters compare by effort and mission size is displayed Figure 2. The cluster number which matches the clusters listed above is on the horizontal axis. The vertical axis is the software development effort. The size of the bubble/circle corresponds to the total mission cost. Here it can be seen that the software effort range overlaps between clusters as what makes them similar is not driven by development effort but by the system characteristics. The main drivers in the cluster formation are Flight Computer Redundancy (Single String vs Dual String (Cold,Warm)), and software size. The median of the range of the effort is approximately 2 to 1 for every cluster except for Cluster 2 which is 10 to 1. The outliers in Cluster 2 are GPM Core as one of the most expensive earth orbiters and

GEMS the least expensive of the earth orbiters. GEMS is likely due to reporting of the development effort, given the project delays. Again, as more data becomes available and we evolve the algorithm we expect these types of issues will go away.

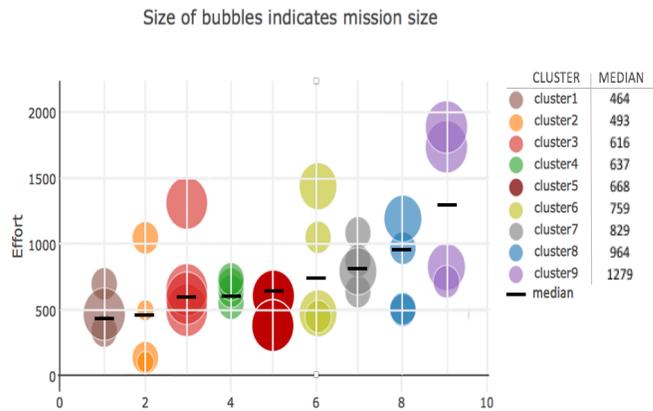


Figure 2 Clusters by effort and mission size

Table 16 Cluster membership and median effort

Cluster Number	Project Name	Effort (WMs) Median
1	Mars Odyssey Maven RBSF	464
2	GPM Core GLORY NuStar GEMS	493
3	GRO GLL Cassini GOES-R	616
4	Stardust Genesis Dawn Messenger	637
5	MRO OSIRIS REX	668
6	Deep Impact JUNO New Horizons Kepler	759
7	MPF Phoenix Grail SMAP	829
8	LRO OCO SDO TIMED	964
9	MER MSL InSight Stereo	1279

*B. Regression Results*

The regression models were developed with cost data from

the CADRe Part C with minimal normalization, partly as a test to see to what extent the raw CADRe data could be successfully used to develop a basic cost model. The additional benefit is that minimal normalization makes verifying the project data used in the model very fast and easy. The basic regression models performed so well that it was decided to include them as part of the ASCoT tool. Again, as with the analogy model the guiding principle was to keep it simple with inputs that can be “approximated” in the early stages of concept development, through Step 1 proposals, and in early phases of the lifecycle. The MRE analysis results are shown below in Table 17 and Figure 3. The results are roughly comparable to the cluster model. The regression models largest errors are smaller than for the analogy model but  $Pred(55) = 66\%$  compared with 90% of the test cases for the analogy model.

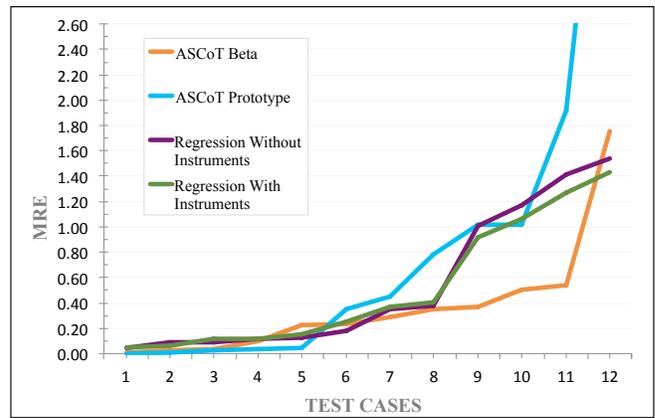


Figure 3 MRE by rank order for all models

Table 17 MRE by Rank Order and Regression Type

MRE by Rank Order and Model Version	
Regression Without Instruments	Regression With Instruments
4%	5%
9%	6%
9%	11%
11%	12%
13%	15%
18%	26%
35%	37%
38%	40%
101%	92%
117%	106%
141%	127%
154%	143%
Median MRE	27%
Mean MRE	54%

The results for all four regression models are shown below (Figures 4-7). As a quick rule of thumb the results indicate that flight software costs around \$4 million at a minimum and then runs 4% of spacecraft cost. This result is heavily driven by the planetary missions as when analyzing only Earth/ Lunar orbiter missions the intercept is not significantly different from 0 and software runs 7% of spacecraft cost.

```
Call:
lm(formula = Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.SW_Cost ~
    Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Total_SC_Cost)

Residuals:
    Min       1Q   Median       3Q      Max
-17456.1  -4040.7   -91.5    3628.8  16570.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.792e+03  1.473e+03   3.253  0.00253 **
Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Total_SC_Cost 4.635e-02  4.929e-03   9.403  4.14e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6638 on 35 degrees of freedom
Multiple R-squared:  0.7164,    Adjusted R-squared:  0.7083
F-statistic: 88.41 on 1 and 35 DF,  p-value: 4.14e-11
```

Figure 4 ALL MISSION TYPES WITHOUT INSTRUMENTS  
 $SW\ Dev\ Cost = 4792 + 0.04635 * (Total\ SC\ Cost, (FY16\$K))$

```

Call:
lm(formula = Total_SW_Dev_Cost_All_Type_Missions$ascot_LR_SW_Cost ~
    Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Total_SC_Cost +
    Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Num_Instr)

Residuals:
    Min       1Q   Median       3Q      Max
-18691.7  -3520.7  -102.7   3765.6  15102.5

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      3.433e+03  2.233e+03   1.537   0.134
Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Total_SC_Cost  4.351e-02  6.060e-03   7.181 2.64e-08 ***
Total_SW_Dev_Cost_All_Type_Missions$ascot_LR.Num_Instr      4.406e+02  5.420e+02   0.813   0.422
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6670 on 34 degrees of freedom
Multiple R-squared:  0.7218,    Adjusted R-squared:  0.7054
F-statistic: 44.11 on 2 and 34 DF,  p-value: 3.579e-10

```

Figure 5 ALL MISSION TYPES WITH INSTRUMENTS  
 $SW\ Dev\ Cost = 3433 + 0.04351 * (Total\ SC\ Cost, (FY16\$K)) + 440.6 * (Num\ of\ Instr)$

```

Call:
lm(formula = Total_SW_Dev_Cost_Earth_LunarOrbiter$SW_Cost ~ Total_SW_Dev_Cost_Earth_LunarOrbiter$Total_SC_Cost)

Residuals:
    Min       1Q   Median       3Q      Max
-9478.6  -2994.6  -554.1   4246.8  7112.1

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.264e+03  1.834e+03   0.689   0.501
Total_SW_Dev_Cost_Earth_LunarOrbiter$Total_SC_Cost  7.162e-02  1.167e-02   6.137 1.43e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4673 on 16 degrees of freedom
Multiple R-squared:  0.7019,    Adjusted R-squared:  0.6832
F-statistic: 37.67 on 1 and 16 DF,  p-value: 1.43e-05

```

Figure 6 EARTH/ LUNAR ORBITER MISSION WITHOUT INSTRUMENTS  
 $SW\ Dev\ COST = 1264 + 0.07162 * (Total\_SC\_Cost, (FY16\$K))$

```

Call:
lm(formula = ascot_LR_1_DeepSpace$SW_Cost ~ ascot_LR_1_DeepSpace$Total_SC_Cost)

Residuals:
    Min       1Q   Median       3Q      Max
-17482  -3324    605    2956   16806

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      5.985e+03  3.122e+03   1.917  0.0759 .
ascot_LR_1_DeepSpace$Total_SC_Cost  4.279e-02  7.462e-03   5.734 5.17e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8658 on 14 degrees of freedom
Multiple R-squared:  0.7014,    Adjusted R-squared:  0.68
F-statistic: 32.88 on 1 and 14 DF,  p-value: 5.166e-05

```

Figure 7 DEEP SPACE MISSION WITHOUT INSTRUMENTS  
 $SW\ Dev\ Cost = 5985 + 0.04279 * (Total\_SC\_Cost (FY16\$K))$

## 5. OVERVIEW OF ASCoT: ANALOGY SOFTWARE COST TOOL

ASCoT Beta will be released sometime in the first or second quarter of calendar year 2017 through the NASA ONCE Model portal. This will be the first official release of the web-based tool for general use and is a major rewrite of the prototype version. The original

ASCoT Prototype was modified from a multi-instance desktop application implemented entirely in Excel to a single-instance web service tool backed by a relational database. In a multi-instance tool it is virtually impossible to propagate updates to the data or the algorithm through all the copies various users have because each instance of the tool has its own data and its own code. Using a web-

based tool, in which both the data and the algorithm are stored remotely, circumvents this problem. All updates to the data or the algorithm occur on the server, so no user is working on a stale version of the tool.

The new data model includes all the mission data on which the algorithm runs, as well as configuration options to allow admin users to work with subsets of the data in their analysis and testing. The same input categories (See Appendix A) are available on the new tool, and steps have been made to prepare the algorithm to accept other inputs.

The front end has been updated to present the cluster results in both tabular and graphical forms with a cleaner user interface.

The updated ASCoT Beta is implemented in Python, Django, MYSQL, and Javascript. The tool was deployed on standard Linux servers, which run CentOS. The graphing library used is plotly.js, which is open source and fully local. These technologies were chosen because they are widely used, open source, and have been used together before by NASA JPL, so sufficient expertise exists to update and maintain the codebase. During early 2017, the ASCoT team will be migrating from their beta environment at JPL to a standard NASA shared server which will support deployment on ONCE.

The main benefits of the web-based implementation are:

1. A single source of data, curated by the tool admins, on which to run the algorithm instead of the disparate versions that are possible on a passed-around Excel tool.
2. Configuration options that filter the data automatically instead of by editing an Excel input file.
3. A cleaner and more expressive user interface that is maintainable and extensible.

The ASCoT architectural design is displayed in Figure 8. The parts that are currently implemented in the model are shown in blue with a dark red border. The team plan is to deliver all capabilities by March 2017.

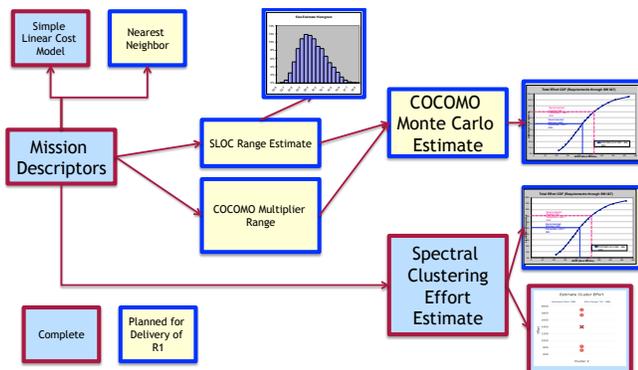


Figure 8 ASCoT Architecture Design

The ASCoT web tool requires the approved user to input their Username and Password as shown in Figure 9

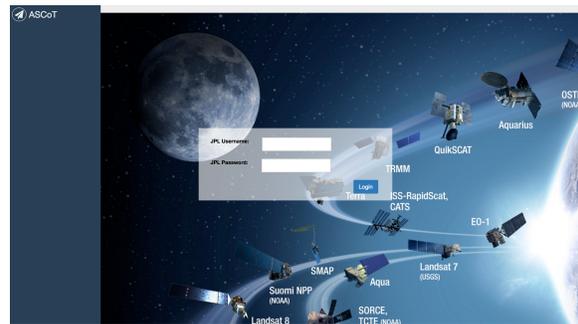


Figure 9 ASCoT web tool log-in page

Users provide values for as many of the following input categories as are known: software size, inheritance levels, mission type, secondary element type, number of instruments, flight computer redundancy type, and total number of deployables. The inputs are defined in detail in Appendix A. Using these inputs, the spectral clustering model finds the cluster in which these inputs best fit and calculates an associated effort estimate.

The “User Estimate” model input and home page shown in Figure 10 consists of three parts: 1) the web tool navigation (left pane); 2) Create New Estimate by the user (center); 3) summary of the User Estimates resulting output by clustered mission names and estimated predicted effort with the range of low, median, and high.

In an example called “Test\_Case\_2”, the ASCoT web tool “Cluster Parameter Variation” in Figure 10 shows the best estimated cluster set based on the user’s defined input and matched to the Nearest Neighbor and cluster sets. One can see that the user’s estimate (indicated by the broken dashed red colored line) is very similar and almost identical based on the parameter for one of the mission in the cluster set. The output will also map the user’s defined parameters to see where it fits with all other clusters. Cluster #1 shows the user’s input (again, indicated by the broken dashed red colored line) compared to other missions in another set of cluster which does not fit well based on how closely the dashed red line traces over the other missions in that cluster.

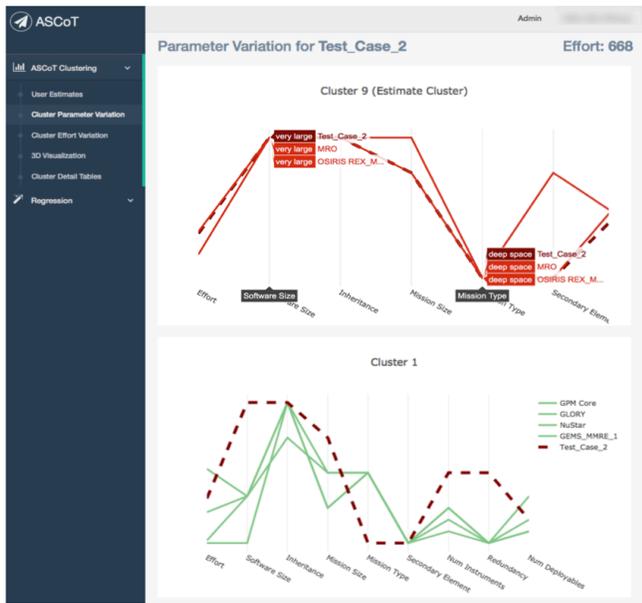


Figure 10 ASCoT Cluster Parameter Variation

“Cluster Effort Variation” Figure 11 shows a bubble chart of all the clusters. The bubble size scales with the level of effort, for example, the larger bubble sets in cluster set #3 has larger effort than in cluster set #2.

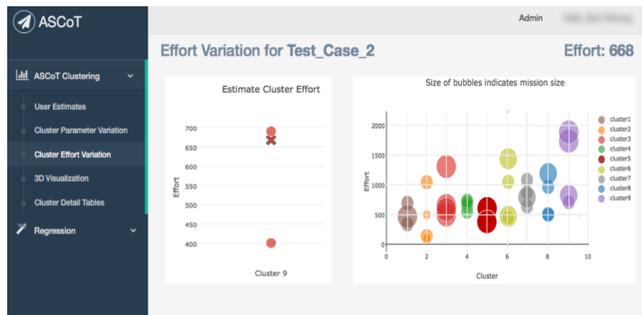


Figure 11 Cluster Effort Variation Bubble Chart

The Regression Analysis is another part of the tool that estimates the Software Development Cost. This is independent from the ASCoT clustering analysis that estimates the effort. The Regression Analysis is a quick and high level estimating tool. There are a total of four regression models, of which three require Total Spacecraft Cost as an input; and the last of which requires Total Spacecraft Cost and Total Number of Instruments as inputs. The first three regression models as shown in Figure 12 generate estimates for 1) All Types of Mission; 2) Planetary Mission Only; and 3) Earth/Lunar Orbiting Missions. The output shows that Total Spacecraft Cost predicting the Total Software Development Cost in FY16\$M.

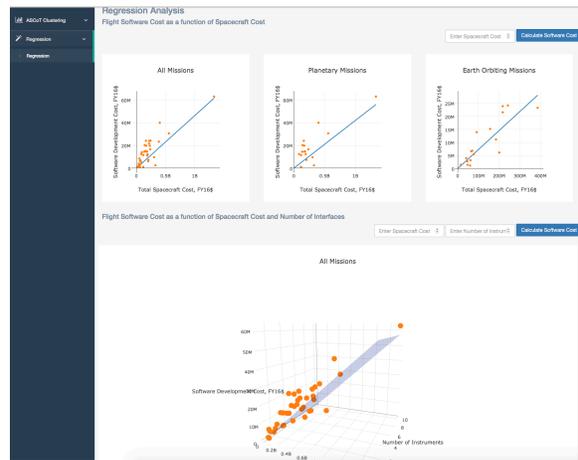


Figure 12 ASCoT Regression Graph Plot

The orange dots are static data points of actual mission sets used to form the regression equation. The User’s input will be indicated by the blue colored oversized circle as shown in Figure 13.

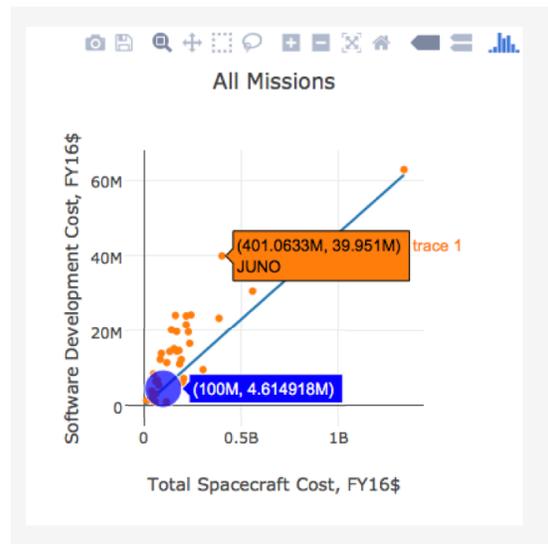


Figure 13 ASCoT Regression Plot

The user can hover over each data point to see the Total Spacecraft and estimated Software Development Costs. The last regression model, which takes in Total Spacecraft Cost and Total Number of Instruments, outputs the regression and estimated result in the 3D view. The axis are estimated Software Development Cost, Total Spacecraft Cost, and Total Number of Instrument. Like the 2D plots, the orange dots are also static data points which missions can be shown by hovering over the dots as shown in Figure 14.

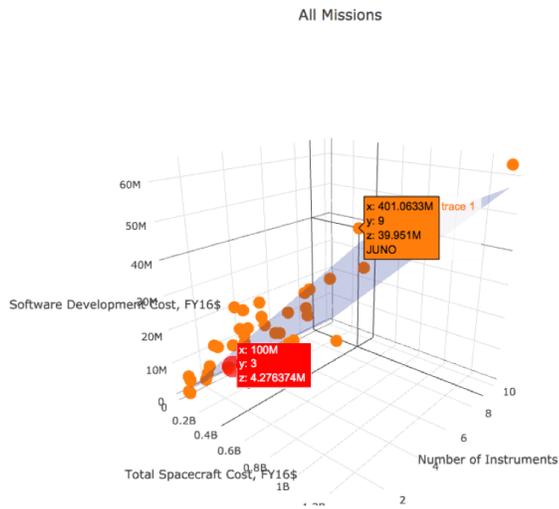


Figure 14 ASCoT 3D Regression Plot

## 5. NEXT STEPS

In the future the team plans to expand and improve the model in a number of different ways. First, the online web model will be deployed to a broader NASA audience using existing NASA shared servers. This will allow the team to collect feedback and also test the process for pushing model updates remotely for updated observations or recalculated clusters. During this time the team also expects to improve the visualizations and user interface. Second, the model will be expanded to include coverage for other types of NASA software, such as Ground systems software. Data collection and data analysis will be required, and the initial capabilities for MMRE testing will be used to select optimal modeling methodologies for the new software types. Lastly, the team hopes to leverage the online web model and apply the methodology development capabilities to other noisy and sparse datasets, such as NASA small satellites (cubesats).

## ACKNOWLEDGEMENT

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

## REFERENCES

- [1] J. Hihn, T. Menzies, L. Juster, G. Mathew, J. Johnson, Improving and Expanding NASA Software Cost Estimation Methods, 2016 IEEE Aerospace Conference, Big Sky, Mt., March, 2016.
- [2] Tim Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes. Validation methods for calibrating software effort models. In Proceedings, ICSE, 2005. Available from <http://menzies.us/pdf/04coconut.pdf>.
- [3] Menzies, T. Chen Z., Port, D., Hihn, J., Simple Software Cost Analysis: safe or Unsafe?, ACM SIGSOFT Software Engineering Notes (SIGSOFT) 30(4)1-6, s2005.

- [4] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. Selecting best practices for effort estimation. IEEE Transactions on Software Engineering, November 2006. Available from <http://menzies.us/pdf/06coseekmo.pdf>.

- [5] "Stable Rankings for Different Effort Models" by Tim Menzies and Omid Jalali and Jairus Hihn and Dan Baker and Karen Lum. Automated Software Engineering December 2010 .

- [6] E. Kocaguneli, T. Menzies, and J.W. Keung. On the value of ensemble effort estimation. Software Engineering, IEEE Transactions on, 38(6):1403–1416, Nov 2012.

- [7] ) B.Boehm, Software Engineering Economics, Prentice Hall, 1981.

## BIOGRAPHY



*software and mission level cost estimation support to JPL's and NASA since 1988.*

**Jairus Hihn** (PhD University of Maryland, 1980) He is a principal member of the engineering staff at the Jet Propulsion Laboratory, and is currently the leading a laboratory wide cost improvement task. He has been developing estimation models and providing



*developing aerospace engineering analysis models and serves as TeamXc's subsystems chair supporting NASA and JPL's spaceflight projects and programs.*

**Michael Saing** (BS, Cal State University, Long Beach). He completed his Aerospace Engineering undergraduate studies and gained his early career work experience at the NASA Ames Research Center. He is currently a Systems Engineer at the Jet Propulsion Laboratory



**Elinor Huntington** is a graduate student at Cal Poly Pomona, studying Computer Science. She works part time at JPL in the Office of Formulation. In a past academic life, she studied Russian Literature.



**James Johnson** is responsible for providing Cost Estimates and Assessments, Schedule Estimates and Assessments, Risk Analyses, and Joint Cost Schedule Risk Analysis for the OCFO Strategic Investments Division (SID) at NASA Headquarters. His work for NASA HQ includes supporting high level Agency

studies, providing support and consultation to projects, and developing policy and guidance for the Agency in the areas of cost, schedule, and risk assessments.



**Tim Menzies** (Ph.D., UNSW, 1995) is a full Professor in CS at North Carolina State University where he teaches software engineering and automated software engineering. His research relates to synergies between human and artificial intelligence, with particular application

to data mining for software engineering.



**George V Mathew** is a graduate student pursuing MS in Computer Science from North Carolina State University. He received his B.Tech degree in Electronics and Instrumentation Engineering from Amrita

University in India. His research interests lies in Software Effort Estimation, Optimizing Requirements Engineering Models and Distributed Multi Objective Optimization. He has also interned as a Software Engineer at Facebook.

## APPENDIX A: COCOMO Model Inputs

	Definition	Low-end = {1,2}	Medium = {3,4}	High-end = {5,6}
<b>Scale factors:</b>				
Flex	development flexibility	development process rigorously defined	some guidelines, which can be relaxed	only general goals defined
Pmat	process maturity	CMM level 1	CMM level 3	CMM level 5
Prec	precedentedness	we have never built this kind of software before	somewhat new	thoroughly familiar
Resl	architecture or risk resolution	few interfaces defined or few risks eliminated	most interfaces defined or most risks eliminated	all interfaces defined or all risks eliminated
Team	team cohesion	very difficult interactions	basically co-operative	seamless interactions
<b>Effort multipliers</b>				
acap	analyst capability	worst 35%	35% - 90%	best 10%
aexp	applications experience	2 months	1 year	6 years
cplx	product complexity	e.g. simple read/write statements	e.g. use of simple interface widgets	e.g. performance-critical embedded systems
data	database size (DB bytes/SLOC)	10	100	1000
docu	documentation	many life-cycle phases not documented		extensive reporting for each life-cycle phase
llex	language and tool-set experience	2 months	1 year	6 years
pcap	programmer capability	worst 15%	55%	best 10%
pcon	personnel continuity (% turnover per year)	48%	12%	3%
plex	platform experience	2 months	1 year	6 years
pvol	platform volatility (frequency of major changes)	$\frac{12 \text{ months}}{1 \text{ month}}$	$\frac{6 \text{ months}}{2 \text{ weeks}}$	$\frac{2 \text{ weeks}}{2 \text{ days}}$
rely	required reliability	errors are slight inconvenience	errors are easily recoverable	errors can risk human life
ruse	required reuse	none	multiple program	multiple product lines
sced	dictated development schedule	deadlines moved to 75% of the original estimate	no change	deadlines moved back to 160% of original estimate
site	multi-site development	some contact: phone, mail	some email	interactive multi-media
stor	required % of available RAM	N/A	50%	95%
time	required % of available CPU	N/A	50%	95%
tool	use of software tools	edit,code,debug		integrated with life cycle

## Appendix B: System Parameters with Definitions and Examples

System Descriptors		
Mission Type	Values	Description
	Earth/Lunar Orbiter	Robotic spacecraft that orbit the earth or moon conducting science measurements. These spacecraft are very similar if not identical to the many commercial satellites used for communication as well as many military satellites. They often can have high heritage and even use production line buses from industry.
	Observatory	Observatories are space based telescopes that support space based astronomy across a wide set of frequencies. They can be earth orbiters or earth trailing at the various Lagrange points created by the gravity fields of the earth, sun and moon.
	Deep Space	Any robotic spacecraft that goes beyond the moons orbit. So this category includes any mission whose destination is a planet, planetoids, any planetary satellite, comet, asteroid or the sun. These mission can be orbiters or flybys or a mixture of both.
	Static Lander	A robotic spacecraft that does its science in-situ or from the surface of a solar system body. It does not move from its original location.
	Rover	A robotic spacecraft that does its science in-situ or from the surface of a solar system body and has the ability to move on the surface. To date all rovers have wheels but in the future they may crawl, walk or hop.
<b>Secondary Element</b>	<b>Values</b>	<b>Description</b>
	None	No secondary element
	Impactor/Probe	A simple impactor with little or no guidance and navigation capability and once released it simply transmits data from its instruments A moderate-complexity impactor which may receive commands after separation, may have some internal guidance control, and several moderately complex instruments.
	Entry Descent and Landing (EDL)	EDL can be simple with a ballistic trajectory or complex with precision landing and hazard avoidance. All landers and Rovers will have an EDL element.
	Sample return	A simple sample return is a like a simple probe but returning to earth. A complex sample return would be a return from a planet surface and requires an ascent stage.
	Multiple Spacecraft	Mission has more than one spacecraft and second spacecraft is a duplicate of the first.
<b>Number of Instruments</b>	<b>Values</b>	<b>Description</b>
	Number of Instruments	Total number of unique instruments on spacecraft.
<b>Number of Deployables</b>	<b>Values</b>	<b>Description</b>
	Number of Deployables	Total number of unique deployables controlled by spacecraft.
		<b>Example</b>
		Number of deployable Solar arrays, booms, robotic arms, etc. Data ranges from 0 to 10 deployments. Median is 3 deployables.
		<b>Example</b>
		Data ranges from 1 to 11 instruments. Median is 4 instruments.
		<b>Example</b>
		Mars Pathfinder is an example of a simple EDL. MSL is an example of a complex EDL
		Stardust is an example of a simple sample return
		Stereo
		<b>Example</b>
		Data ranges from 1 to 11 instruments. Median is 4 instruments.
		<b>Example</b>
		Number of deployable Solar arrays, booms, robotic arms, etc. Data ranges from 0 to 10 deployments. Median is 3 deployables.
		<b>Example</b>
		Mars Exploration Rover (MER)
		<b>Example</b>
		Most orbiters
		Cassini-Huygens was a simple probe. Deep Impact had a medium complexity probe.



**Appendix B: System Parameters with Definitions and Examples (Continued)**

<b>Flight Computer Redundancy</b>	<b>Values</b>	<b>Description</b>	<b>Example</b>
	Single String	Spacecraft has no redundancy in the flight computer	Most Earth Orbiters
	Dual String - Cold backup	Spacecraft has redundant flight computers. Backup is normally off, is powered up and boots when prime string goes down	Most Deep space missions
	Dual String - Warm backup	Backup computer is powered on and monitoring state of prime computer, but does not need to maintain continuous operation (e.g., a sequence may be restarted, attitude control restarts with last known state, etc.)	MSL
<b>Software Delivered Code</b>	<b>Values</b>	<b>Description</b>	<b>Example</b>
	Small	Delivered logical lines of code is < 50 KSLOC	Small earth orbiters
	Medium	Delivered logical lines of code is < 50 KSLOC and < 120 KSLOC	LRO, Kepler
	Large	Delivered logical lines of code is < 120 KSLOC and < 220 KSLOC	LCROSS, SMAP, Phoenix
	Very Large	Delivered logical lines of code is > 220 KSLOC	Rovers
<b>Inheritance</b>	<b>Values</b>	<b>Description</b>	<b>Example</b>
	Low to None	Total Inherited code, including modified code is < 10% of delivered code.	MER, TIMED, LRO
	Low	Total Inherited code, including modified code is between 10% to 20% of delivered code.	Deep Impact, New Horizons
	Medium	Total Inherited code, including modified code is >= 20% and < 50% of delivered code.	Messenger, MRO
	High	Total Inherited code, including modified code is >= 50% and < 80% of delivered code.	JUNO, SDO, GPM core
	Very High	Total Inherited code, including modified code is a minimum of 80% of delivered code.	MAVEN, Grail, NOAA-N-Prime
<b>Total Mission cost</b>	<b>Values</b>	<b>Description</b>	<b>Example</b>
	Small	Total Mission cost including operations in FY15 dollars is > \$120M and < \$220 million	Wise, small earth orbiters
	Medium	Total Mission cost including operations in FY15 dollars is > \$220 million and < \$600 million	Discovery class missions
	Large	Total Mission cost including operations in FY15 dollars is > \$600 million and < \$1.1 billion	New Frontiers class missions
	Very Large	Total Mission cost including operations in FY15 dollars is > \$1.1 billion	Large assigned mission, MSL

