

Improving Software Size Estimates by Using Probabilistic Pairwise Comparison Matrices¹

Dr. Jairus Hihn
California Institute of Technology
Jet Propulsion Laboratory
Jairus.M.Hihn@jpl.nasa.gov

Karen T. Lum
California Institute of Technology
Jet Propulsion Laboratory
Karen.T.Lum@jpl.nasa.gov

Abstract

The Pairwise Comparison technique is a general purpose estimation approach for capturing expert judgment. This approach can be generalized to a probabilistic version using Monte Carlo methods to produce estimates of size distributions. The probabilistic pairwise comparison technique enables the estimator to systematically incorporate both estimation uncertainty as well as any uncertainty that arises from using multiple historical analogies as reference modules. In addition to describing the methodology, the results of the case study are also included. This paper is an extension of the work presented in [Lum & Hihn, 2003] and will show how the original software size estimates compared to the actual delivery size. It will also describe the techniques used to modify the approach based on lessons learned.

1. Introduction

Software cost estimates are typically required in the early stages of the life-cycle when requirements and design specifications are immature. Under these conditions, the production of an accurate cost estimate requires extensive use of expert judgment and the quantification of significant estimation uncertainty. Research has shown that under the right conditions, expert judgment can yield relatively “accurate” estimates [Hihn & Habib-agahi, 1990]. Unfortunately, most expert judgment-based estimates do not meet these conditions and frequently degenerate into outright guessing. At its best, expert judgment is a disciplined combination of a ‘best guess’ and historical analogies. Pairwise comparison matrices provides a formal, systematic means of extracting, combining, and capturing expert judgments and their relationship to analogous reference data [Saaty, 1980].

The use of matrices of pairwise comparisons as an approach for deriving a cardinal ranking vector from subjective paired comparisons was first introduced by Saaty in 1977 as part of the Analytical Hierarchy Process (AHP) [Saaty, 1977]. AHP, as originally proposed, is a decision-making or prioritization technique. The usefulness of this technique for software size estimation was recognized in the mid-eighties [Bozoki, 1986; Lambert, 1986]. Bozoki provided a more detailed description of his approach in [Bozoki, 1993]. Using a pairwise comparison matrix to estimate software size requires an expert’s judgment as to each module’s *relative* bigness compared to one another. The effectiveness of this approach is supported by experiments that indicate that the human mind is better at identifying relative differences than at estimating absolute values [Shepperd, 2001, Miranda, 1999]. In this paper, for ease of exposition, pairwise comparison matrices will be called judgment matrices following [Crawford, 1987].

An opportunity to apply the pairwise comparison technique arose when a cost estimate was required for a mission critical ground software project. The software team members wanted to be more rigorous in how they approached the estimating task. They also imposed a number of constraints. The team wanted to estimate the size and cost of its next software development task based on a recently completed development activity. Since there were numerous sources of potential risk and uncertainty associated with the next delivery, the technical staff wanted to provide ranges for the ratio comparisons in order to visualize the actual probability distribution for the estimates. In addition, they had multiple sources of historical software size data that could be incorporated. While more information is generally considered helpful, this particular situation presented some complexities that the standard

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

approach did not address well when incorporating uncertainty. Therefore, the pairwise technique had to be adapted to address distributional inputs in the context of multiple reference analogies.

A description of how this technique was applied to effort estimation in combination with software cost models is provided in [Lum, Hihn, 2003]. This paper describes the approach and algorithms used to generalize the paired ratio comparison matrix technique in order to utilize information inherent in multiple estimates, multiple reference projects, and estimator range information for the purpose of generating estimated size distributions. In addition, this paper details the application of this technique in the context of an actual software project, in order to assess its effectiveness.

2. Pairwise Comparison Technique

Creating a judgment matrix involves creating an $n \times n$ matrix ($\mathbf{A}^{n \times n} = [a_{ij}]$), where n is the number of entities (for software, this could be modules, use cases, requirements, etc.), being compared. Each element, a_{ij} , in the matrix is an estimate of the relative size of entity i with respect to entity j , that is $\frac{Size_i}{Size_j}$. The properties of a judgment matrix

require that elements be:

- (1) reciprocal, $a_{ij} = 1/a_{ji}$, which means that entity i is a_{ij} times bigger than entity j , then entity j is $1/a_{ij}$ times smaller than entity i ;
- (2) the same size as itself, which means that all diagonal elements $a_{ii} = 1$.

The implication of these properties is that only the upper or lower triangle of the judgment matrix must be filled in. For example, see Table 1, which is a judgment matrix with estimates of the relative software size of four modules. The values in Table 1 indicate that Module 1 is two times as big as Module 2, three times bigger than Module 3, and four times bigger than Module 4, and so forth. Note that there is no a priori reason that all the values in the upper triangle are greater than 1.

Table 1. Example judgment matrix

	Module 1	Module 2	Module 3	Module 4
Module 1		2	3	4
Module 2			1.5	2
Module 3				1.5
Module 4				

Based on Conditions 1 and 2 above, the matrix can be completed as follows:

Table 2. Example completed judgment matrix

	Module 1	Module 2	Module 3	Module 4
Module 1	1	2	3	4
Module 2	0.5	1	1.5	2
Module 3	0.33	0.67	1	1.5
Module 4	0.25	0.5	0.67	1

One way to interpret the judgment matrix is that each column yields a different ranking vector for the purpose of determining the relative size of the four entities. Each vector is normalized such that the module that corresponds to itself (the diagonal elements) is always 1, and it is the reference module against which all comparisons in the same column are made. Therefore, column 1 indicates that module 2 is half as big as module 1; module 3 is 33% of the size of module 1; and module 4 is 25% of module 4. Each column can be interpreted in this manner. In Table 2, there are four different rankings (an $n \times n$ matrix yields n independent ranking vectors). In a case where different ranking vectors have different rank orders for each entity, there is more estimation uncertainty around the entities being compared.

A special case exists when a judgment matrix is perfectly consistent. This occurs when $a_{ij} \times a_{jk} = a_{ik}$ for all i, j, k . If a judgment matrix is consistent, then each vector is equivalent to all the others, or each vector can be transformed into the other via a linear transformation. This means that there is really one vector of unique information. Time has been wasted in making all of these pairwise comparisons in the example, since only four

numbers needed to be guessed and not six. Fortunately, it turns out that judgments are rarely consistent, unless the estimator is cheating.

More frequently, inconsistent matrixes will result. The example in Table 2, gives four different rankings each yielding a slightly different set of estimates for the modules. The benefit is that there is plenty of information from which a final estimate can be generated. However, only one vector is desired, not n vectors.

There have been a number of mathematical procedures proposed for deriving a single ranking vector from an inconsistent judgment matrix. These produce numbers that meet the conditions of a ratio scale. This means the slope of a line has been defined but the intercept or origin of the scale is still unknown. Therefore, the actual sizes of the modules are unknown, and only their relative sizes are known. However, as long as at least one of the modules used to derive judgment matrix is an historical analogy, that module can be used to determine the intercept or origin to determine the estimated sizes for each module.

The original approach proposed by [Saaty, 1977] was to use the Perron-Frobenius right eigenvector. Research has shown, however, that this is one of the worst techniques to use [Hihn & Johnson, 1988]. There are many potential solutions to this problem. The Geometric Mean method, which is very easy to calculate, has been advocated by many authors for various reasons. [Hihn & Johnson, 1988; Crawford and Williams, 1985]. Miranda chose to use the geometric mean procedure because of its simplicity and the results achieved in during experimentation with thirty participants [Miranda, 2001]. Therefore, the geometric mean is the recommended approach used in this case.

The Geometric Mean is calculated as $v_i = \prod_{j=1}^n a_{ij}^{1/n}$, which yields a vector

$$\mathbf{V} = \begin{bmatrix} v_1 \\ v_2 \\ v_{\dots} \\ v_n \end{bmatrix}$$

that meets the requirements of a ratio scale. The example in Table 2 would yield the vector

$$\begin{bmatrix} 2.21 \\ 1.11 \\ 0.76 \\ 0.54 \end{bmatrix}$$

As an example, the 2.21 is derived from $(1 \times 2 \times 3 \times 4)^{1/4}$.

Once the ratio scale vector is calculated, the size of each known entity can be calculated using at least one known historical analogy to normalize the vector, and in essence define the origin for the ratio scale. This allows one to convert the vector to cardinal numbers yielding absolute values of the size estimates. The size of at least one of the elements in the ratio scale is needed as a reference to derive a multiplier m

$$m = \frac{Size_{ref}}{v_{ref}}, \text{ } ref \text{ is one of the modules } i \text{ through } n$$

which is used to calculate the size of the other elements. The formula is as follows:

$$\mathbf{S} = \begin{bmatrix} Size_1 \\ Size_2 \\ Size_{\dots} \\ Size_n \end{bmatrix} = m \times \mathbf{V} = \begin{bmatrix} m \times v_1 \\ m \times v_2 \\ m \times v_{\dots} \\ m \times v_n \end{bmatrix}$$

Using the example in Table 2 and assuming the reference module is v_3 at 2000 lines of code, this step would result in the following:

$$m = 2000/0.76 = 2632.15$$

$$Size_1 = 2632.15 \times 2.21 = 5826$$

$$Size_2 = 2632.15 \times 1.11 = 2913$$

$$Size_3 = 2632.15 \times 0.76 = 2000$$

$$Size_4 = 2632.15 \times 0.54 = 1414$$

In summary an estimate of size using pairwise comparison matrices can be generated using four steps:

- 1) Estimate the relative size of all modules
- 2) Derive the judgment matrix
- 3) Compute the geometric mean across each row in the matrix
- 4) Derive size estimate by normalizing values to the reference module

3. Probabilistic Pairwise Comparison Technique

Two major adaptations to the basic pairwise comparison technique described above were made: (1) the incorporation of distributions for pairwise judgments and (2) the use of multiple reference modules. There are many distributions that could be used. A log normal distribution would make it possible to derive a closed form solution. However, it is difficult for engineers to estimate the mean and variance of a log normal distribution. The use of estimation ranges (low, mode, and high) was found to be a more practical approach. Since it is important for a software manager and engineers to have a clear understanding of their inputs and estimates, the simplest distribution cognitively is the triangular distribution. That is,

$$a_{ij} \sim TriPDF(min, mode, max)$$

where the element a_{ij} is a triangular distribution $TriPDF$ with a minimum variate (min), a peak variate ($mode$), and a maximum variate (max). This requires the use of a Monte Carlo technique to combine the different distributions, but with modern computers and software that is not a difficult task.

To illustrate the technique the example from the previous section will be used. The first step is that the subjective judgments in Table 1 can be entered as distributions as shown in Table 3. Here are three elements as distributions and three as point values.

Table 3. a_{ij} as distributions

	Module 1	Module 2	Module 3	Module 4
Module 1		2	3	$TriPDF(3, 4, 6)$
Module 2			1.5	$TriPDF(2, 2.5, 3)$
Module 3				$TriPDF(1, 1.5, 3)$
Module 4				

In Table 3, element a_{14} and element a_{34} are entered as distributions with $a_{14} \sim TriPDF(3, 4, 6)$ and $a_{34} \sim TriPDF(1, 1.5, 3)$, respectively. The element a_{14} would be interpreted as Module 1 is most likely 4 times bigger than Module 4, but could be as much as 6 times bigger, or at a minimum, it could be the three times the size. The element a_{34} would be interpreted as Module 3 is most likely 1.5 times bigger than Module 4, but could be as much as 3 times bigger, or at a minimum, it could be the same size. Random draws are made from these distributions to determine the geometric mean vector, which becomes:

$$\mathbf{V}_{PDF} = \left[v_{PDFi} = \prod_{j=1}^n a_{ij}^{1/n} \right], \text{ where } a_{ij} \sim TriPDF(min, mode, max).$$

The result is that each element in the geometric mean vector is now a distribution v_{PDFi} .

Another major adjustment to the basic pairwise technique is the manner in which multiple reference software modules are incorporated. The basic method described above works well for the case where there is a single reference analogy. However, having multiple reference analogies with the typical case of an inconsistent judgment matrix creates a dilemma. A different total size estimate is generated depending upon which reference module is used. The solution proposed here involves incorporating the multiple references by capturing the different possible reference values as a distribution. This way the basic Monte Carlo structure that has been set up here can be used as a general purpose approach for any type of ranking problem.

A triangular distribution is used to capture the differences between the multiple reference-derived multipliers m_{PDFref_i} through m_{PDFref_x}

$$m_{PDFref_i} = \frac{Size_{ref_i}}{V_{PDFref_i}}$$

If the matrix were consistent, having multiple references should theoretically result in the same multiplier value

$$m_{PDF}^* = m_{PDFi} = m_{PDFx}$$

A triangular distribution is merely a simple method of capturing the multiple references. Therefore, one solves for a single distribution of the multipliers m_{PDF}^*

$$m_{PDF}^* \sim TriPDF(\minMultiplier, GMmultiplier, \maxMultiplier)$$

where

$$\minMultiplier = \min_{i=1}^x m_{PDFref_i}$$

$$GMmultiplier = \left(\prod_{i=1}^x (m_{PDFref_i}) \right)^{1/x}$$

$$\maxMultiplier = \max_{i=1}^x m_{PDFref_i}$$

for i through x number of references.

For example, the first four columns in Table 4 are the completed judgment matrix shown in Table 3 based on $a_{ij} \sim TriPDF(\min, mode, \max)$. The gray shaded cells of the matrix are completed using the properties of a judgment matrix. Therefore, elements a_{11} and a_{22} are the same size as itself, 1. Element a_{42} is the reciprocal of element a_{24} , which is

$$\frac{1}{TriPDF(2, 2.5, 3)}$$

The remainder of the judgment matrix was completed in a similar manner.

Table 4. Completed matrix and multipliers based on distributional inputs²

	Module 1	Module 2	Module 3	Module 4	Geometric Mean PDF (V_{PDFi})	References ($Size_{refi}$)	Multiplier PDF ($Size_{refi}/V_{PDFrefi}$)	Size Estimate PDF ($Size_i \times (m_{PDF}^*)$)
Module 1	1	2	3	4.33	2.26	6000	2657.1	
Module 2	0.5	1	1.5	2.5	1.17	3000	2563.72	
Module 3	0.33	0.67	1	1.83	0.80	2000	2503.36	
Module 4	0.23	0.4	0.545	1	0.47			???

Using the algorithm described above yields the geometric mean distribution function, whose mean values are shown in the Geometric Mean PDF column of Table 4,

$$\begin{bmatrix} 2.26 \\ 1.17 \\ 0.80 \\ 0.47 \end{bmatrix}$$

² The means of the distributions are shown in the cells.

Having multiple references and an inconsistent judgment matrix results in many multipliers of different values. For example, given that the actual sizes of Modules 1, 2, and 3 are 6000, 3000, and 2000 lines of code respectively, and using the average values shown in Table 4, the expected multipliers would be

$$Expected(m_{PDF_{ref_1}}) = Size_{ref_1} \div v_{PDF_{ref_1}} = 6000 \div 2.26 = 2657.1,$$

$$Expected(m_{PDF_{ref_2}}) = Size_{ref_2} \div v_{PDF_{ref_2}} = 3000 \div 1.17 = 2563.72, \text{ and}$$

$$Expected(m_{PDF_{ref_3}}) = Size_{ref_3} \div v_{PDF_{ref_3}} = 2000 \div 0.80 = 2503.36.$$

If the multiplier derived from Module 1 is selected, the expected size of Module 4 would be

$$Expected(Size_4) = 2657.1 \times 0.47 = 1259,$$

while if the multiplier derived from Module 3 is used, the expected size of Module 4 would be

$$Expected(Size_4) = 2503.36 \times 0.47 = 1186.$$

In this example, having multiple references results in three different size estimates. A problem arises in deciding which multiplier should be used to estimate the remaining unknown modules.

To derive the size estimate of Module 4, the triangular distribution of the minimum multiplier, geometric mean of the multipliers as a mode, and maximum multiplier is used. This produces a distribution for m_{PDF}^* as shown in Figure 1.

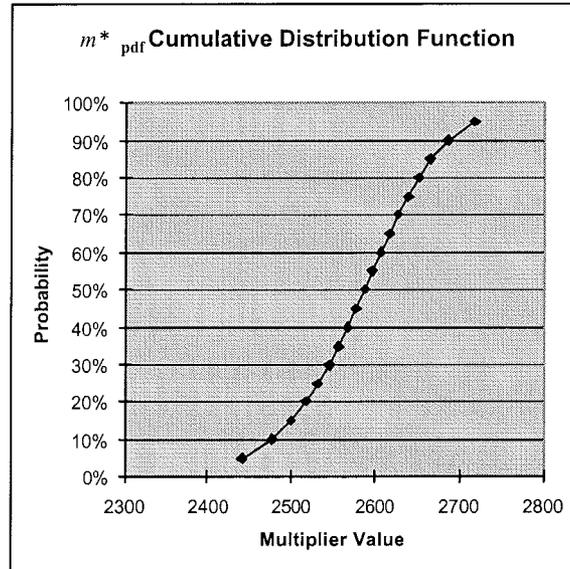


Figure 1. Cumulative distribution function of m_{PDF}^*

Therefore, the size of Module 4 is

$$Size_{PDF_4} = m_{PDF}^* \times v_{PDF_4},$$

which is a distribution that can be shown as a cumulative probability curve (Figure 2).

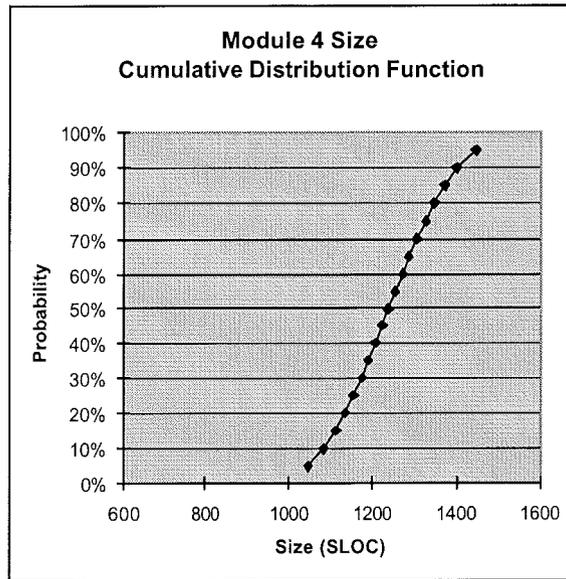


Figure 2. Module 4 size cumulative distribution function based on Table 3 inputs

4. Application

The techniques described above were applied to the estimation of a mission critical ground software delivery at JPL. The project wished to estimate the size and cost of its next delivery (Delivery 2) based on analogy to its current delivery (Delivery 1). However, many actuals were not recorded during the current delivery and therefore needed to be reconstructed. As it was easier to reconstruct effort data, the software team first estimated the direct cost of Delivery 2 through pairwise comparison to delivery 1 using effort as the size factor. The direct effort estimate resulted in an estimate much lower than the project team had expected [Lum and Hihn, 2003]. The project team was then convinced to attempt an estimate using a cost model. Since cost models require size as an input, the Probabilistic Pairwise Comparison technique was used to estimate the size of Delivery 2. The Delivery 1 source code was run through a code counter, and the pairwise technique was then repeated for estimating the size of Delivery 2 using the modules of Delivery 1 as reference points.

A major step in obtaining acceptance from the team was allowing them to think of the software in terms of capabilities. This was very important since their cognitive model was based on the concept of capabilities and the team had no experience in estimating software size by lines of code or function points. Since the pairwise comparison technique is based on estimating the relative size differences between modules, it was assumed that this relative difference would be the same - subject to a linear transformation. This is a critical assumption, which will be discussed in more detail at the end of the paper.

The first delivery consisted of one major software function (Function A) that could be further subdivided into five modules. The second delivery consisted of two major software functions (Functions B and C) that each had five modules. Code counts for the user interface modules of each Function were not available, as they were in a language for which a code counter was not available³. Therefore only four modules of each function were compared with each other. The four modules for each second delivery software function were compared with the four modules of the first delivery software function based on their relative size. Function B and Function C were not compared with each other, because the two software functions would be developed by different people who found it difficult to compare functions that had not been developed yet. Therefore the pairwise judgment matrix of this exercise produced two 8 x 8 matrices with 64 possible comparisons per matrix. (Figure 3 and Figure 4).

³ The cost of the user interfaces for Function B and Function C were estimated in a different manner, using direct effort comparisons, since size references were not available for the user interface module of Function A.

A significant adaptation to the paired comparisons method described by others is that the estimators were allowed to give ranges for

$$a_{ij} = \frac{Size_i}{Size_j}, \text{ where } a_{ij} \sim TriPDF(\min, mode, max).$$

These ranges were easily captured and a Monte Carlo distribution of the ratio scales was easily implemented in Microsoft Excel®.

		Function B (Delivery 2)				Function A (Delivery 1)			
		Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI
Function B	Core	1		TriPDF(5, 75, 1)	TriPDF(5, 75, 1)	0.2		2	1.5
	Mapping		1	TriPDF(25, 375, 5)	TriPDF(5, 75, 1)	0.2	1	1	0.75
	Stochastics			1	TriPDF(1, 1.5, 2)	TriPDF(25, 625, 1)	TriPDF(1, 2, 3)	TriPDF(5, 1.25, 2)	TriPDF(2, 5, 4)
	Smoother				1	TriPDF(25, 375, 5)	TriPDF(1, 1.5, 2)	TriPDF(25, 375, 5)	TriPDF(5, 75, 1)
Function A	Force Models					1			
	Event Based Integration						1		
	Partial Derivatives							1	
	Core Integration								1

Figure 3. Original comparison inputs for Function B vs. Function A modules

		Function C (Delivery 2)				Function A (Delivery 1)				
		MM	DM	FF	Stations	FM	EBI	PD	CI	
Function C	Measurement Models	1		TriPDF(3, 5, 3)	2	TriPDF(2, 2, 3)	TriPDF(0.5, 0.7, 1)	TriPDF(2, 2, 2.5)	TriPDF(2, 2, 2.5)	TriPDF(1, 1, 1.5)
	Delay Models		1	TriPDF(1.5, 0.625, 0.75)	0.5	TriPDF(0.2, 0.25, 0.5)	TriPDF(1, 1, 25, 1.5)	TriPDF(0.7, 0.75, 1)	TriPDF(0.75, 0.875, 1)	
	File Formats			1	TriPDF(1, 25, 1, 375, 1.5)	TriPDF(0.4, 0.45, 0.5)	TriPDF(1, 1, 25, 1.5)	TriPDF(1, 1, 25, 1.5)	TriPDF(0.75, 0.875, 1)	
	Stations				1	TriPDF(0.4, 0.45, 0.5)	TriPDF(0.5, 0.75, 1)	TriPDF(1, 1, 25, 1.5)	TriPDF(0.75, 1, 25)	
Function A	Force Models					1				
	Event Based Integration						1			
	Partial Derivatives							1		
	Core Integration								1	

Figure 4. Original comparison inputs for Function C vs. Function A modules

Quadrants II and III (Figure 5) of the matrices were easily completed utilizing the judgment matrix properties: $a_{ij}=1/a_{ji}$ and $a_{ii} = 1$. Since Delivery 1 was completed and all the actuals were known (Table 5), Quadrant IV was derived with the actuals as reference points. For example, it is known from actuals that the development size of force models was approximately 1.45 times more than event based integration effort and 1.48 times more than the partial derivatives effort, etc. As quadrant IV is based on actuals, it is the only consistent quadrant, satisfying the property $a_{ij} \times a_{jk} = a_{ik}$. Since the other three quadrants are based upon subjective judgments, they are unlikely to be consistent.

The screenshot shows an Excel spreadsheet with a comparison matrix. The matrix is a 10x10 grid with columns labeled A through J and rows labeled 1 through 10. The diagonal elements are 1.00. The matrix is divided into three quadrants: Quadrant II (top-right), Quadrant III (bottom-left), and Quadrant IV (bottom-right). A formula bar at the top shows "=RV.TriSample(RAND(),2,3,4)". A cell in the bottom-right quadrant (row 6, column J) contains the value 3.00.

		Function B (Delivery 2)				Function A (Delivery 1)				
		Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI	
Function B	Core	1.00	1.00	0.75	0.75	0.2		2.00	1.50	
	Mapping		1.00		0.75	0.20	1	1.00	0.75	
	Stochastics			1.33	1.50	0.63	2.00		3.00	
	Smoother			1.33	1.33	1.00	0.30	1.50	0.38	0.75
Function A	Force Models				5.00	2.67	1.00	1.45	1.48	0.66
	Event Based Integration				0.50	0.67	1	1.00	1.02	0.45
	Partial Derivatives				1.00	1.00	0.67	1.48	1.00	0.44
	Core Integration				0.67	1.33	1.52	2.20	2.28	1.00

Figure 5. Random variable formulas are entered in cells for elements with a range of comparisons

The matrix was completed such that a triangular random variable draw formula was entered in the cells for which ranges were given (Figure 5).

Table 5. Actual size of reference modules⁴

	Actual Size (KSLOC)
Force Models	6.5
Event Based Integration	4.5
Partial Derivatives	4.4
Core Integration	10
User Interface	Code count not available

The geometric mean of the rows for the matrix was then computed to arrive at the ratio scale vector (labeled Geometric Mean PDF in Figure 6).

		Function B (Delivery 2)				Function A (Delivery 1)				Geometric Mean PDF	References (SLOC)	Multiplier PDF's to get	Estimate PDF (SLOC)
		Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI				
Function B	Core	1.00	1.00	0.75	0.75	0.20	2.00	1.00	1.50	0.09			3152.199
	Mapping	1.00	1.00	0.20	0.75	0.20	1.00	1.00	0.75	0.07			3972.981
	Stochastics	1.33	2.67	1.00	1.50	0.63	2.00	1.25	3.00	0.15			8324.702
	Smoother	1.33	1.33	0.67	1.00	0.20	1.50	0.33	0.75	0.09			4787.212
Function A	Force Models	5.00	5.00	1.00	2.67	1.00	1.45	1.48	0.66	0.23	6548	32788.4	
	Event Based Integration	0.60	1.00	0.50	0.57	0.10	1.00	1.02	0.45	0.07	4820	61104.7	
	Partial Derivatives	1.00	1.00	0.60	2.57	0.67	0.90	1.00	0.44	0.10	4410	23695.7	
	Core Integrations	0.67	1.33	0.33	1.33	1.52	2.20	3.26	1.00	0.12	3965	81450.2	
											GM Multiplier	51734.2	22737
											Min Multiplier	32788.4	Total Function B Size
											Max Multiplier	81450.2	
											m* per	55334.3	

Figure 6. A single multiplier PDF m_{PDF}^* is derived from the multiple reference multipliers and estimates are a function of m_{PDF}^*

As there were multiple references (four modules from Delivery 1), computing the reference multiplier required extra steps. Since the subjective pairwise judgments were inconsistent in this application, four reference modules produced four different multipliers, each producing significantly different estimates. A Monte Carlo run on the triangular random draw of the four multipliers using the lowest value multiplier as a minimum, the geometric mean of the four multipliers as the mode, and the highest value multiplier as a maximum was performed:

$$m_{PDF}^* \sim TriPDF(\minMultiplier, GMmultiplier, \maxMultiplier).$$

An estimate of each Delivery 2 module was then calculated using the new randomly drawn multiplier m_{PDF}^* .

Allowing ranges in the pairwise judgments and drawing from the four possible multipliers captures the uncertainty in the estimate and serves to average out estimation errors. Utilizing a Monte Carlo technique produces size cumulative distribution functions for each module of the Delivery 2 functions, each Delivery 2 function, and the total Delivery 2 size. Table 6 and Table 7 show the resulting distributions of each element in Function B and Function C respectively, as well as the total size of each function in delivery 2.

Table 6. Function B size estimates

	Core	Mapping	Smoother	Stochastics	Function B Total SLOC
5th Percentile	3621	2800	3316	5880	15898
Mode	5740	3831	4985	10374	22211
95th percentile	6796	5230	6444	11878	29950

⁴ Code Counts derived using Galorath's Count95 tool.

Table 7. Function C size estimates

	Measurement Models	Delay Models	File Formats	Stations	Function C Total SLOC
5th Percentile	8209	3100	4525	3976	19866
Mode	10737	3789	5971	4833	24356
95th percentile	12837	4897	7053	6241	30899

5. Results

Delivery 2 was completed in nine months and within 10% of the budget with 24,506 lines of code. Not all of the original capabilities were included in the release. The task leads subjectively estimated the differences between the planned module capabilities and the capabilities actually delivered, which varied from 0 to 100%, with most modules at 90% of the plan. While Function B delivered 90% of its capabilities, Function C delivered significantly reduced capability than planned. Almost none of the Delay Models functionality was delivered. Only 50% of Measurement Models and 90% of File Format capabilities delivered. The size estimates were adjusted for the differences in planned functionality and are compared with the actual sizes in Table 8.

Functions B and C were collectively overestimated by 44%. The team's estimate also produced a very wide range across the modules. On the surface, this appears to be a mediocre performance, but compared to an average size underestimate of 70% on JPL Deep Space Network subsystems upgrades (from 1989 to 1997) and underestimates in excess of several hundred percent on recent JPL flight software systems, it is a marked improvement, especially considering that the team had virtually no experience in estimating size by lines of code.

Table 8. Functions B and C delivered software size vs. adjusted size estimates

		FUNCTION B					FUNCTION C				
		Core	Mapping	Smoother	Stochastics	Total SLOC	Measurement Models	Delay Models	File Formats	Stations	Total SLOC
Actual Sizes		5035	6115	244	748	12142	1959	689	7611	2105	12364
Adjusted Estimates*	5th Percentile	3259	2520	2984	5292	14308	4105	0	4073	3976	12187
	Mode	5166	3448	4487	9337	19990	5369	0	5374	4833	14976
	95th Percentile	6116	4707	5800	10690	26955	6419	0	6348	6241	18928
	% Deviation from Mode	3%	-44%	1739%	1148%	65%	174%	-100%	-29%	130%	21%

*Original estimates were adjusted to reflect the actual percentage of functionality delivered.

In addition, it is necessary to take consider that the team estimated modules by subjectively comparing relative differences in capability - not lines of code. The two modules, which are off by an order of magnitude, are Smoother and Stochastics in Function B. These two modules are very algorithmically intensive, such that six months was spent in deriving and implementing the algorithms for 200-800 lines of code. If these two algorithmically intensive elements are excluded from the analysis, the total size estimate is within 30% of the actual (Table 9). This result strongly suggests that pairwise comparisons between capabilities can be used to estimate relative differences in size for procedural code but not algorithmically intensive code. This result is not particularly surprising, as a number of authors have also noted problems with the function point sizing metric when estimating algorithmically intensive code.

Table 9. Function B actual size vs. adjusted size estimate excluding algorithmic elements

		FUNCTION B		
		Core	Mapping	Total SLOC**
Actual Sizes		5035	6115	11150
Adjusted Estimates*	5th Percentile	3259	2520	5791
	Mode	5166	3448	8200
	95th Percentile	6116	4707	10810
	% Deviation from Mode	3%	-44%	-26%

*Original estimates were adjusted to reflect the actual percentage of functionality delivered.

** Excluding the algorithmically-intensive Stochastics and Smoother elements.

6. Conclusion

The pairwise comparison technique is a general purpose estimation approach for capturing expert judgment and can be relatively easily implemented using Microsoft Excel® if the geometric mean method is used to derive the ratio vector V . In this document, it has been documented how this approach can be further expanded into a probabilistic version using Monte Carlo methods in order to produce estimates of size distributions. The probabilistic pairwise comparison technique enables the estimator to systematically incorporate both estimation uncertainty as well as any uncertainty that arises from using multiple historical analogies as reference modules. While more work needs to be done to verify the effectiveness of this approach, it does appear that it can be used to improve software size estimates for non-algorithmically intensive code.

One natural extension of this analysis is that if all pairwise comparisons are assumed to be log normally distributed, a relatively simple closed form solution exists for deriving the estimated size distributions as a function of the means. Standard deviations of the pairwise comparisons and the Monte Carlo computations would not be necessary. The disadvantage of this approach is that it has been found in practice to be easier for engineers to estimate low, most likely and a high than to subjectively estimate a mean and variance of a distribution, especially if it is skewed, as in a log normal distribution.

The most important result of this study may be sociological in nature. The study proved to be a relatively successful example of introducing quantitative techniques into a skeptical software development. Working with the technical staff in a way that they preferred and that was consistent with their cognitive models allowed us to engage them, eventually leading to a relatively rigorous size estimate and use of a cost model.

7. References

- [1] Boehm, B., et al., *Software Cost Estimation with COCOMO II*, Upper Saddle River, New Jersey, Prentice Hall PTR: 2000.
- [2] Bozoki, G. "Software Size Estimator (SSE)," Centre National d'Etudes Spatiales (CNES), Toulouse, France, June 1986.
- [3] Bozoki, G. "An Expert Judgment-Based Software Sizing Model," *Journal of Parametrics*, Volume XIII, Number 1, May 1993.
- [4] Crawford, G. "The Geometric Mean Procedure for Estimating the Scale of a Judgment Matrix," *Mathematical Modelling* 9/3-5, 327-334.ing 9/3-5, 327-334.
- [5] Crawford, G. and Williams, C "The Analysis of Subjective Judgment Matrices," Rand Corporation, R-2572-1-AF, May 1985. A Project AIR FORCE report prepared for the USAF.
- [6] Hihn, J.M. and Habih-agahi, H. "Cost Estimation of Software Intensive Projects: A Survey of Current Practices," *Proceedings of the Thirteenth IEEE International Conference on Software Engineering*, May 13-16, 1991. (also SSORCE/EEA Report No. 2. August 1990)
- [7] Hihn, J.M. and Johnson, C. "Evaluation Techniques for Paired Ratio Comparison Matrices in a Hierarchical Decision Model," *Measurement in Economics*, Physical-Verlag Heidelberg, 1988.
- [8] Lambert, J. "A Software Sizing Model," *Journal of Parametrics*, Vol. Vi, 1986, pp75-87.
- [9] Lum, K., and Hihn, J., *Estimation of Software Size and Effort Distributions Using Paired Ratio Comparison Matrices*, Proceedings of the 3rd Annual Joint Conference of the International Society of Parametric Analysts (ISPA) and Society of Cost Analysis and Estimation (SCEA), 17-20 June, 2003, Orlando, FL.
- [10] Miranda, E.: "Establishing Software Size Using the Paired Comparisons Method." Proc. of the IWSM'99, Lac Superieur, Quebec, Canada, September 1999, pp. 132-142
- [11] Miranda, E. "Improving Subjective Estimates Using Paired Comparisons," *IEEE Software* Jan/Feb 2001.
- [12] Saaty, T. "A Scaling method for Priorities in a Hierarchical Structure". *J. Math. Psychology* Vol. 15 1977, p 234-281.
- [13] Saaty, T. *The Analytic Hierarchy Process*, McGraw-Hill, New York, NY: 1980.
- [14] Shepperd, M. and Cartwright M. "Predicting with Sparse Data," *IEEE Transactions on Software Engineering*, Nov. 2001, Vol. 27, No. 11.