

**Estimation of Software Size and Effort Distributions
Using Paired Ratio Comparison Matrices
Karen Lum
Jairus Hihn, Ph.d.**

California Institute of Technology/Jet Propulsion Laboratory
m/s 301-180
4800 Oak Grove Drive
Pasadena, CA 91109

Abstract

Recently the Software Quality Improvement (SQI) Project has been formed at NASA's Jet Propulsion Laboratory (JPL). SQI will enable and promote software best practices, and leverage JPL experience in software engineering in support of major software projects, throughout the entire software life-cycle. The goal of the SQI Project is to establish an operational software improvement program that results in the continuous measurable improvement at JPL. Its objectives include improving cost and schedule predictability, improving the quality of mission-critical software, reducing software defect rates during testing and operations, increasing software development productivity, promoting software reuse, and reducing project start-up time. In this paper we will document one of several approaches being introduced at JPL to improve its ability to improve cost estimation accuracy early in the project life-cycle.

We were recently confronted with a problem where a cost estimate was required for a piece of mission critical software. The technical staff did not trust cost models, and they had numerous sources of potential risk and uncertainty associated with the next build. The task wanted to estimate the costs of its next delivery based on the cost of its current delivery. However, the task had virtually no retrievable historical data from their previous builds, for either effort or software size. To help them formalize their expert judgment-based estimates, we considered using Galorath Corporation's SEER-SSM, which is an adaptation of the paired ratio comparison matrixes described in Saaty's Analytical Hierarchy Process. While this approach is very attractive, a number of problems arose in that we had multiple reference projects, the technical staff wanted to provide ranges for the ratio comparisons, and we wanted a distribution for the size and/or effort estimates rather than a point value. We also had to use the method to help us reconstruct the actuals as well as estimate the next build.

In this paper we describe how 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. We document how this approach can be further generalized to a probabilistic version using Monte Carlo methods to produce estimates of size and effort 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, we will also describe the results of the case study.

Estimation of Software Size and Effort Distributions Using Paired Ratio Comparison Matrices¹

**Karen Lum
Jairus Hihn, Ph.d.**

California Institute of Technology/Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Introduction

All too frequently, a software cost estimate is required in the early stages of the life-cycle when requirements and design specifications are immature. To produce a cost estimate under these conditions requires extensive use of expert judgment and addressing significant estimation uncertainty. Research has shown that expert judgment under the right conditions 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. Especially, when the estimate is made without the assistance of a cost estimation professional. At its best, expert judgment is a disciplined combination of a best guess combined with historical analogies. Using the method of paired-ratio comparison matrices provides a formal systematic way to extract, combine, and capture expert judgments and how they relate to analogous reference data [Saaty, 1980]. A version of which is supported by the SEER-SSM (Software Sizing Model) tool, which is primarily advertised as a tool for estimating software size but can be used to estimate virtually anything requiring expert judgment [Bozoki, 1993].

We were recently confronted with such a problem when a cost estimate was required for a piece of mission critical ground software. In this case, we were not providing an independent estimate but assisting the software team to generate its own estimates. The software team members wanted to be more rigorous in how they approached the estimating task but they also imposed a number of constraints. The team wanted to estimate the costs of its next software development task based on the cost of a recently completed development activity. However, the task had virtually no retrievable historical data from its previous deliveries, for either effort or software size. Because of this we also had to use the method to help us reconstruct the actuals as well as estimate the cost of the next delivery. While it was possible to obtain code counts on completed code, there was little record of inheritance and furthermore the technical staff did not trust costing by software size or using cost models. To make matters more complicated they had numerous sources of potential risk and uncertainty associated with the next delivery.

To help them formalize their expert judgment-based estimates, we considered using Galorath Corporation’s SEER-SSM. While this approach is very attractive, a number of problems arose. The interface required that they provide too many inputs. The technical staff also wanted to see the entire judgment matrix, to make direct multiplicative comparisons, to provide ranges for the ratio comparisons, and we wanted the actual probability distribution for the size and/or effort estimates. This paper describes the approach and algorithms used to generalize the paired ratio

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.

comparison matrix technique to use information inherent in multiple estimates, multiple reference projects, and estimator range information to generate estimated effort and size distributions. In addition to describing the methodology, we will also describe the results of the case study.

Pairwise Comparison Technique

The use of matrices of paired-ratio 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 power of this technique for prediction and specifically software size estimation was recognized in the mid-eighties [Bozoki, 1986; Lambert, 1986]. Using a paired-ratio comparison matrix to estimate software size or effort 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, paired-ratio comparison matrices will be called judgment matrices following [Crawford, 1987].

Creating a judgment matrix involves creating an $n \times n$ matrix ($A^{n \times n} = [a_{ij}]$), 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 we will assume is 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, four times bigger than Module 3, and three 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	4	3
Module 2			1.5	2
Module 3				2
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	4	3
Module 2	0.5	1	1.5	2
Module 3	0.25	0.67	1	2
Module 4	0.33	0.5	0.5	1

One way to interpret the judgment matrix is that each column yields a different ranking vector for 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. Hence, column 1 indicates that module 2 is half as big as module 1, module 3 is 25% of the size of module 1, and module 4 is 33% of module 4. Each column can be interpreted in this manner. In Table 2, we have generated four different rankings (an $n \times n$ matrix yields n independent ranking vectors).

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 and that we wasted our time making all of these pairwise comparisons, as we could have just guessed four numbers and not six. Fortunately, it turns out that judgments are rarely consistent, unless the estimator is cheating.

More frequently, inconsistent matrixes will result, such as in Table 2, which gives us four different rankings each yielding a slightly different set of estimates for the modules. The good news is that we have lots of information from which we can generate our final estimate, and the bad news is that we do not want n vectors, we want one vector.

There have been a number of mathematical procedures proposed for deriving a single ranking vector from an inconsistent judgment matrix. What these produce are numbers that meet the conditions of a ratio scale. This means we have defined the slope of a line but we do not know the intercept or origin of the scale. Hence we do not know the actual sizes of the modules only their relative sizes. 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 allowing us 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 chooses to use the geometric mean procedure because of its simplicity and the results achieved in experiments he ran with thirty participants [Miranda, 2001]. Therefore, we used and recommend the use of the geometric mean.

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 $(1 \times 2 \times 4 \times 3)^{1/4}$.

Once the ratio scale vector is calculated, the size or effort 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, which allows us to convert the vector to cardinal numbers yielding absolute values of the size or effort estimates. The size or effort 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 or effort 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:

$$\begin{aligned} m &= 2000/0.76 = 2632 \\ Size_1 &= 2632 \times 2.21 = 5816 \\ Size_2 &= 2632 \times 1.11 = 2922 \\ Size_3 &= 2632 \times 0.76 = 2000 \\ Size_4 &= 2632 \times 0.54 = 1421 \end{aligned}$$

In summary an estimate of size or effort 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/effort estimate by normalizing values to the reference module

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, we have found that it is difficult for engineers to estimate the mean and variance of a log normal distribution. It is much easier for them to estimate ranges or a low, mode, and high. Since we are working closely with a software manager and engineers when using this method they need to have a clear understanding of how their inputs/estimates are used. Therefore, the simplest distribution cognitively is the triangular distribution. That is,

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

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 significant drawback.

To illustrate the technique we will use the example from the previous section. The first step is that the subjective judgments in Table 1 can be entered as distributions as shown in Table 3. Here we show four elements as distributions and two remain as point values.

Table 3. a_{ij} as Distributions

	Module 1	Module 2	Module 3	Module 4
Module 1		<i>TriPDF</i> (1, 2, 3)	4	<i>TriPDF</i> (1, 3, 4)
Module 2			<i>TriPDF</i> (1, 1.5, 3)	<i>TriPDF</i> (1, 2, 3)
Module 3				2
Module 4				

In Table 3, element a_{12} and element a_{14} are entered as distributions with $a_{12} \sim \text{TriPDF}(1, 2, 3)$ and $a_{14} \sim \text{TriPDF}(1, 3, 4)$, respectively. The element a_{12} would be interpreted as Module 1 is most likely 2 times bigger than Module 2, but could be as much as 3 times bigger, or at a minimum, it could be the same size. The element a_{14} would be interpreted as Module 1 is most likely 3 times bigger than Module 4, but could be as much as 4 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:

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

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

Another major adjustment to the basic pairwise technique is the manner by which we incorporate the use of multiple reference software modules. 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 the dilemma that a different total size estimate is generated depending upon which reference module is used. The solution proposed here is to incorporate 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 can be used as a general purpose approach.

We use a triangular distribution to capture the differences between the multiple reference-derived multipliers $m_{PDF\ ref_i}$ through $m_{PDF\ ref_x}$

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

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

$$m_{PDF}^* = m_{PDF\ i} = m_{PDF_n}$$

A triangular distribution is merely a simple method of capturing the multiple references.

Therefore, we solve for a single distribution of the multipliers m_{PDF}^*

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

where

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

$$GMmultiplier = \prod_{i=1}^x (m_{PDF\ ref_i})^{1/x}$$

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

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_{21} is the reciprocal of element a_{12} , which is

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

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	4	2.67	2.15	6000	2791.8	
Module 2	0.5	1	1.83	2	1.16	3000	2578.2	
Module 3				2	0.72	2000	2767.6	
Module 4					0.55			???

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,

$$\begin{bmatrix} 2.15 \\ 1.16 \\ 0.72 \\ 0.55 \end{bmatrix}$$

If we have multiple references and an inconsistent judgment matrix, we would get 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

$$\begin{aligned} \text{Expected}(m_{PDF\ ref_1}) &= Size_{ref_1} \div v_{PDF\ ref_1} = 6000 \div 2.15 = 2791.8, \\ \text{Expected}(m_{PDF\ ref_2}) &= Size_{ref_2} \div v_{PDF\ ref_2} = 3000 \div 1.16 = 2578.2, \text{ and} \\ \text{Expected}(m_{PDF\ ref_3}) &= Size_{ref_3} \div v_{PDF\ ref_3} = 2000 \div 0.72 = 2767.6. \end{aligned}$$

If we use the multiplier derived from Module 2, the expected size of Module 4 would be

$$\text{Expected}(Size_4) = 2578.2 \times 0.55 = 1418,$$

while if we use the multiplier derived from Module 3, the expected size of Module 4 would be

$$\text{Expected}(Size_4) = 2767.6 \times 0.55 = 1522.2.$$

In this example, having multiple references causes us to get three different size estimates. Which multiplier do we use to estimate the remaining unknown modules?

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

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

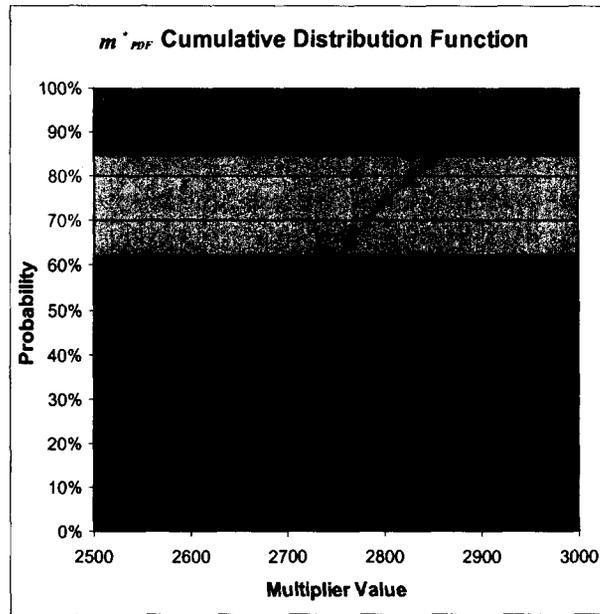


Figure 1. Cumulative Distribution Function of m_{PDF}^*

Therefore, the size of Module 4 is

$$Size_{PDF4} = m_{PDF}^* \times v_{PDF4},$$

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

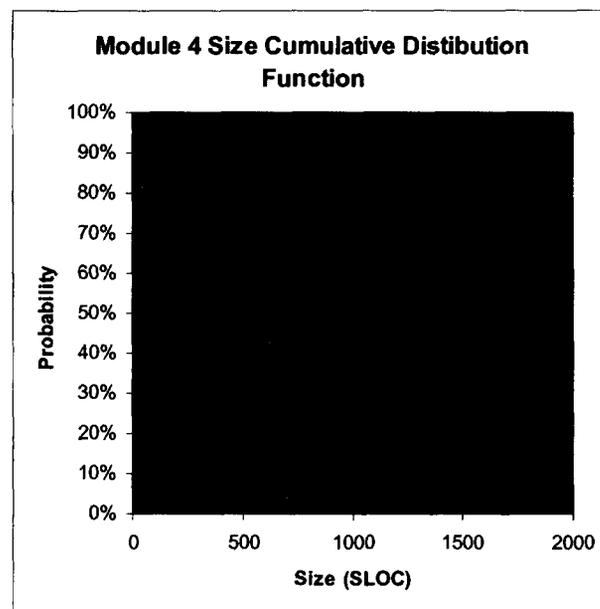


Figure 2. Module 4 Size Cumulative Distribution Function Based on Table 3 Inputs

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 costs of its next delivery (delivery 2) based on the cost of its current delivery (delivery 1). However, many actuals were not recorded during the current delivery and had to be reconstructed. Lines of code data were not available on the delivery 1. However, it was much easier to reconstruct the amount of effort expended. The effort data of delivery 1 was reconstructed through dialogue with the project team members. Therefore, we used the pairwise comparison technique to estimate direct effort for delivery 2.

The first delivery consisted of one major software function (Function A) that could be further subdivided into five modules. The second delivery – the delivery for which a cost estimate was needed – consisted of two major software functions (Functions B and C) that each had five modules. The five modules for each second delivery software function were compared with the five modules of the first delivery software function based on their relative bigness of effort. The software modules of the second delivery 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, we have two 10 x 10 matrices (Figure 3 and Figure 4). This gives 100 possible comparisons per matrix.

A significant adaptation to the paired comparisons method described by others is that we allowed the estimators 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)				
		UI	Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI	UI
Function B	User Interface	1	TriPDF(4,4,5.5)	TriPDF(4,4,5.5)	TriPDF(2,2,5.3)	TriPDF(3,3,5.4)	1.2	TriPDF(7,7,5.8)	4	3	TriPDF(5,5,7)
	Core		1	1	TriPDF(.5, .75, 1)	TriPDF(.5, .75, 1)	0.2	2	1	1.5	TriPDF(1.5, 1.75, 2)
	Mapping			1	TriPDF(.25, .375, .5)	TriPDF(.5, .75, 1)	0.2	1	1	0.75	0.5
	Stochastics				1	TriPDF(1, 1.5, 2)	TriPDF(.25, .625, 1)	TriPDF(1, 2, 3)	TriPDF(.5, 1.25, 2)	TriPDF(2, 3, 4)	TriPDF(1, 2, 3)
	Smoother					1	TriPDF(.25, .375, .5)	TriPDF(1, 1.5, 2)	TriPDF(.25, .375, .5)	TriPDF(.5, .75, 1)	TriPDF(.5, .75, 1)
Function A	Force Models						1				
	Event Based Integration							1			
	Partial Derivatives								1		
	Core Integration									1	
	User Interface										1

Figure 3. Original Comparison Inputs for Function B vs. Function A modules

		Function C (Delivery 2)					Function A (Delivery 1)				
		MM	DM	UI	FF	Stations	FM	EBI	PD	CI	UI
Function C	Measurement Models	1	TriPDF(2,2,5.3)	TriPDF(2,2,5.3)		2	TriPDF(0.5, 0.7, 1)	TriPDF(2, 2.25, 2.5)	TriPDF(2, 2.25, 2.5)	TriPDF(1.5, 1.75, 2)	TriPDF(2, 2.5, 3)
	Delay Models		1	1	TriPDF(0.5, 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)	1.5
	User Interface			1	TriPDF(0.5, 0.625, 0.75)	TriPDF(0.5, 0.625, 0.75)	TriPDF(0.2, 0.225, 0.25)	TriPDF(0.5, 0.7, 1)	TriPDF(0.5, 0.7, 1)	TriPDF(0.5, 0.625, 0.75)	TriPDF(1, 1.125, 1.25)
	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)	2
	Stations					1	TriPDF(0.4, 0.45, 0.5)	TriPDF(0.5, 0.75, 1)	TriPDF(1, 1.25, 1.5)	TriPDF(0.75, 1.1, 1.25)	TriPDF(1, 1.25, 1.5)
Function A	Force Models						1				
	Event Based Integration							1			
	Partial Derivatives								1		
	Core Integration									1	
	User Interface										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, Quadrant IV was derived with the actuals³ as reference points. For example, we know from actuals (Table 5), that the development effort of force models was 9 times more than event based integration effort and 2.25 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 matrix was completed such that a triangular random variable draw formula was entered in the cells for which ranges were given (Figure 5).

		Function B (Delivery 2)					Function A (Delivery 1)				
		UI	Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI	UI
Function B	User Interface	1.00	4.50	4.50	2.50	3.50	1.20	7.50	4.00	3.00	6.00
	Core	0.22	1.00	1.00	0.75	0.75	0.20	2.00	1.00	1.50	1.75
	Mapping	0.22	1.00	1.00	0.36	0.75	0.20	1.00	1.00	0.75	0.50
	Stochastics	0.40	1.33	1.00	1.00	1.50	0.63	2.00	1.25	3.00	2.00
	Smoother	0.29	1.33	1.33	0.67	1.00	0.36	1.50	0.36	0.75	0.75
Function A	Force Models	0.83	5.00	5.00	1.60	2.67	1.00	9.00	2.25	2.25	2.25
	Event Based Integration	0.13	0.50	1.00	0.50	0.67	0.11	1.00	0.25	0.25	0.25
	Partial Derivatives	0.25	1.00	1.00	0.80	2.67	0.44	4.00	1.00	1.00	1.00
	Core Integration	0.33	0.67	1.00	0.33	1.33	0.44	4.00	1.00	1.00	1.00
	User Interface	0.17	0.57	2.00	0.50	1.33	0.44	4.00	1.00	1.00	1.00

Figure 5. Random Variable Formulas Are Entered in Cells for Elements with Range Comparisons

Table 5. Actual Effort of Reference Modules

	Actual Effort (Work-Months) ⁴
Force Models	9
Event Based Integration	1
Partial Derivatives	4
Core Integration	4
User Interface	4

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).

³ These “actuals” were not originally documented and were reconstructed after discussion with the project team members.

⁴In addition to the development effort for each module, there was some management and 5 WM of testing for delivery 1 effort. Testing effort for Delivery 2 was treated as a fixed percentage of development effort and added on after the pairwise comparisons for development effort was computed, based on the Delivery 1 actuals. Management for delivery 2 was also treated as a fixed cost, and estimated at 9.6 WM by the project team. The 9.6 WM of management was added on after the pairwise comparisons for development effort was computed.

		Function B (Delivery 2)					Function A (Delivery 1)								
		UI	Core	Mapping	Stochastics	Smoother	FM	EBI	PD	CI	UI	Geometric Mean PDF	References	Multiplier PDFs to get Actuals	Estimate PDF
17															
18	User Interface	1.00	4.50	4.50	2.50	3.50	1.20	7.50	4.00	3.00	6.00	3.21			12.04
19	Core	0.22	1.00	1.00	0.75	0.75	0.20	2.00	1.00	1.50	1.75	0.82			3.07
20	Mapping	0.22	1.00	1.00	0.38	0.75	0.20	1.00	1.00	0.75	0.50	0.58			2.20
21	Stochastics	0.40	1.33	2.67	1.00	1.50	0.63	2.00	1.25	3.00	2.00	1.35			5.07
22	Smoother	0.29	1.33	1.33	0.67	1.00	0.38	1.50	0.38	0.75	0.75	0.73			2.72
23	Force Models	0.83	5.00	5.00	1.60	2.67	1.00	9.00	2.25	2.25	2.25	2.49	9.00	3.62	
24	Event Based Integration	0.13	0.50	1.00	0.50	0.67	0.11	1.00	0.25	0.25	0.25	0.36	1.00	2.76	
25	Partial Derivatives	0.25	1.00	1.00	0.80	2.67	0.44	4.00	1.00	1.00	1.00	0.99	4.00	4.02	
26	Core Integration	0.33	0.67	1.33	0.33	1.33	0.44	4.00	1.00	1.00	1.00	0.96	4.00	4.63	
27	User Interface	0.17	0.57	2.00	0.50	1.33	0.44	4.00	1.00	1.00	1.00	0.86	4.00	4.64	
28													GMmultiplier	3.87	
29													MinMultiplier	2.76	
30													MaxMultiplier	4.64	
31													m**_PDF	3.76	

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 (five modules from delivery 1), computing the reference multiplier required some extra steps. Since the subjective pairwise judgments were inconsistent in this application, having five reference modules produced five different multipliers that would each produce significantly different effort estimates. A Monte Carlo run on the triangular random draw of the five multipliers using the lowest value multiplier as a minimum, the geometric mean of the five 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 five possible multipliers captures the uncertainty in the estimate and averages out estimation errors. Utilizing a Monte Carlo technique produces cumulative distribution functions for each module of the delivery 2 functions, each delivery 2 function, and the total delivery 2 effort.

The original pairwise judgments and the reference judgments did not include effort for testing or management, which the project team considered to be a fixed cost of development. The direct effort cumulative distribution function was adjusted (shifted) to include a fixed percentage of management and testing. After this adjustment, the effort was still lower than the development team had expected. After discussion with the team, we felt that various differences in characteristics between the two deliveries were not captured in the original pairwise judgments, mainly the product complexity. In addition, the development team felt that they were more experienced in the delivery 1 application than the delivery 2 application, which was fairly new to them. An adjustment factor, based on COCOMO II's effort multipliers, was applied:

$$\text{Adjustment Factor} = \frac{EAF_{\text{Delivery2}}}{EAF_{\text{Delivery1}}} = \frac{0.57}{0.42} = 1.36,$$

where *EAF* is the product of the COCOMO II effort multipliers. See [Boehm, 2000] for details. This shifted the estimated cumulative distribution function to a range that was closer to what the project team had expected (Figure 7). However, this direct effort estimate was still lower than the actual budgeted effort.

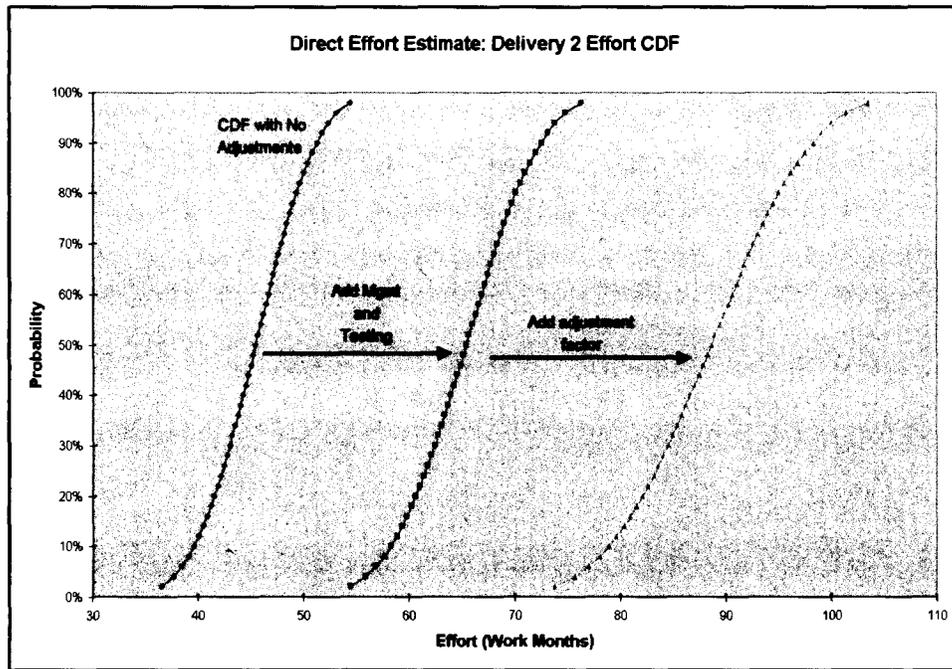


Figure 7. Delivery 2 Direct Effort Cumulative Distribution Function

After seeing the resulting effort cumulative distribution function, we were able to convince the project team to attempt an estimate based on a cost model. 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. Code counts for the user interface of each function were not available as they were in a language for which a code counter was not available. Therefore the pairwise judgment matrix of this exercise produced two 8 x 8 matrices. The effort for the user interfaces of functions B and C (based on the effort derived for the user interfaces in the previous direct effort estimate) were added back after a model-based estimate was run.

The project team members said that the original pairwise judgments inputs from Figure 3 and Figure 4 could be used as the comparisons for this second size-based estimate. This gives reason to question the correctness of the direct effort estimate.

Quadrants II and III of the judgment matrices were completed using the properties described in the previous estimate. Table 6 shows the actual code counts of the delivery 1 functions that were used to derive Quadrant IV of the judgment matrices. The geometric mean vector, multipliers,

and size cumulative size distributions were calculated in a similar manner as described in the direct effort estimate.

Table 6. Actual Size of Reference Modules⁵

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

A “Minimum,” “Mode,” and “Maximum” size estimate was then entered into an Excel-based COCOMO II tool that accepts ranges. The 5th percentile and 95th percentile size estimates for the delivery 2 functions (Functions B and C) were extracted from the size cumulative distribution functions as the “minimum” and “maximum” estimate, respectively (Table 7 and Table 8). The mode size estimates were calculated from the Monte Carlo draw results and used as the “mode” estimate. In addition, there will be inheritance with little to no modification of a subroutine library that would probably be about 5% of the size of the delivery 2 functions. Once the size inputs and COCOMO II cost driver ratings from interviews with the software team members were obtained, an effort cumulative distribution function was output. The cumulative distribution functions were shifted to include the user interfaces.

Table 7. Function B Size Estimates

	Core	Mapping	Smoother	Stochastics	Function B Total SLOC
5th Percentile	3621	2800	3316	5880	15898
Mode	5740	3831	4985	10374	22155
95th percentile	6796	5230	6444	11878	29950

Table 8. 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	24968
95th percentile	12837	4897	7053	6241	30899

⁵ Code Counts derived using Galorath’s Count95 tool.

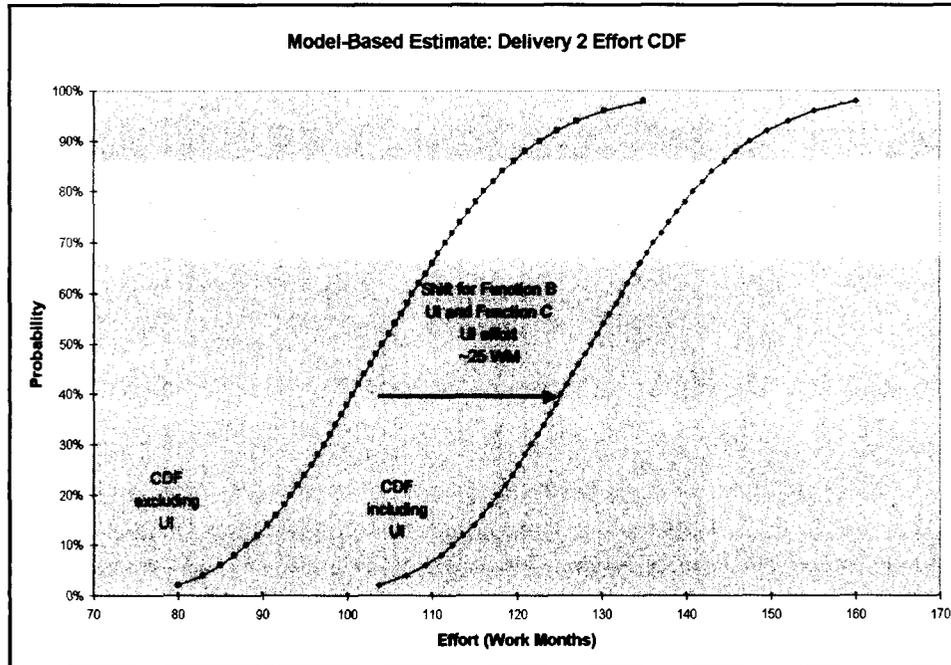


Figure 8. Delivery 2 Model-based Effort Cumulative Distribution Function

The resulting effort cumulative distribution function was higher than the estimate produced by the first pairwise exercise and much closer to what the developers had expected (Figure 8), within the range of their actual budgeted effort.

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 . We have documented and demonstrated how this approach can be further generalized to a probabilistic version using Monte Carlo methods to produce estimates of size and effort 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 the actual application of this technique, we found that even though the software engineers felt very uncomfortable estimating size and preferred to estimate effort directly, a size based COCOMO estimate provided a more realistic estimate based on the current budget and historical experience. This was surprising as an earlier study found that JPL software engineers were much better at estimating effort than size [Hihn and Habib-agahi, 1991]. We suspect that what occurred was that the delivery 1 functionality was typical of software the team had developed over the last few years, was highly modular and relatively low complexity, and was a less formal delivery than for the planned delivery 2 functionality. This meant that for a direct effort estimate, their "minds" were not calibrated correctly, while using a size-based estimate with size actuals from functions they had just completed gave a more accurate size estimate and the cost model properly reflected all the supporting activities and effort multiplier impacts.

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] Miranda, E.: "Establishing Software Size Using the Paired Comparisons Method." Proc. of the IWSM'99, Lac Superieur, Quebec, Canada, September 1999, pp. 132-142
- [10] Miranda, E. "Improving Subjective Estimates Using Paired Comparisons," *IEEE Software* Jan/Feb 2001.
- [11] Saaty, T. "A Scaling method for Priorities in a Hierarchical Structure". *J. Math. Psychology* Vol. 15 1977, p 234-281.
- [12] Saaty, T. *The Analytic Hierarchy Process*, McGraw-Hill, New York, NY: 1980.
- [13] Shepperd, M. and Cartwright M. "Predicting with Sparse Data," *IEEE Transactions on Software Engineering*, Nov. 2001, Vol. 27, No. 11.