Improving Subjective Estimates Using Paired Comparisons

Eduardo Miranda, *Ericsson Research Canada*

Despite the existence of structured methods for software sizing and effort estimation, the so-called "expert" approach seems to be the prevalent way to produce estimates in the software industry. The paired-comparisons method offers a more accurate and precise alternative to "guesstimating."

Most practitioners and project managers still produce estimates based on ad hoc or so-called "expert" approaches, even though several software sizing methods—counting source lines of code,[1] function points,[2] full function points,[3] and object points, to name a few—are well known and have been available for a long time. Among the most common explanations given for not adopting more formal estimation practices are the lack of necessary information at the beginningof the project, the specificity of the domain addressed, the effort and time required, and the need to introduce a vocabulary foreign to stakeholders without a software background.

However, as Figure 1 shows, ad hoc size estimates have problems of their own. Their accuracy (the closeness of a measured value to the true one) and precision (indicating how repeatable a measurement is) leave much to be desired. The problem is not academic: inaccurate size estimates automatically translate into questionable project budgets and schedules.

This article presents a method based on paired comparisons, which social science researchers use for measuring when there is no accepted measurement scale or when a measurement instrument does not exist. Although not new, the idea has received little attention in the literature. Earlier work includes Target Software's software sizing method[4] and more recent articles by Focal Point AB[5] and by Bournemouth University's Empirical Software Engineering Research Group,[6] which uses the analytic hierarchical process to prioritize requirements relative to their cost and estimate effort respectively.[7]

Overall approach

The idea behind paired comparisons is to estimate the size of n entities by asking one or more experts to judge the entities' relative largeness rather than to provide absolute size values. (*Entities* can be requirements, use cases, modules, features, objects, or anything else relevant to all stakeholders and for which it is possible to know the number of lines of code, hours, or any other magnitude that could later be used for planning purposes.) By requiring multiple and explicit decisions about the relative size of every two entities and by using easily available historical

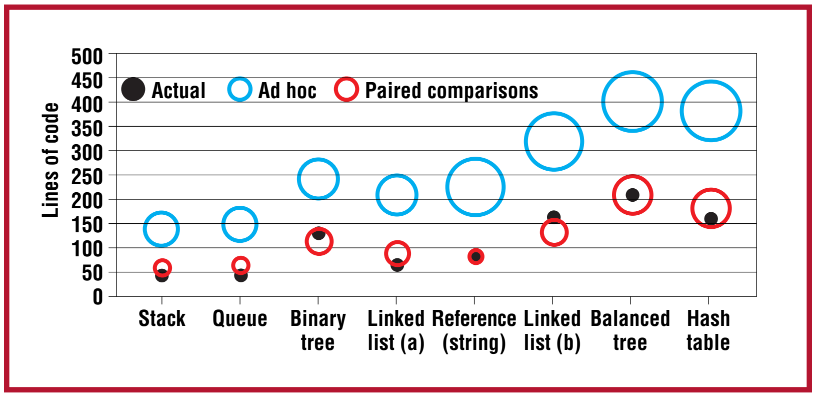


Figure 1. The accuracy and precision of different estimation approaches (results of a study involving over 30 software professionals and graduate students): actual measurements, ad hoc estimates, and paired-comparison estimates.

data—rather than a single comparison to some vague notion of size buried in the estimator's mind—the paired-comparisons method improves both the accuracy and the precision of estimates, as shown in Figure 1. These findings are consistent with the conclusions of Albert L. Lederer and Jayesh Prasad's study, which shows that using historical data and documented comparisons produce better estimates than those based on intuition and guessing.[8]

As Figure 2 shows, with the proposed approach we start by arranging the entities to be sized according to their perceived largeness. We then assess the relative size of each one with respect to all the others and record this information in what is called a *judgment matrix*. From the judgments made, we derive a ratio scale using a simple mathematical procedure and then calculate the absolute size of the entities using the ratio scale and a reference value. Should the need arise, judgments can be reviewed for internal consistency.

The method is independent of the type of entities chosen. It is important, however, that the sizes of the entities being estimated do not differ by more than one order of magnitude, because our ability to accurately discriminate size diminishes as the difference between the entities becomes larger.[7,9,10]

Judgment matrices

A judgment matrix is a square matrix of size n, where n is the number of entities being compared; and each element a_{ij} captures the relative size of entity i with respect to entity j. The elements of the matrix are defined as

$$A^{n\times n}=\left[a_{ij}\right] = \begin{cases} a_{ij}=\dfrac{s_i}{s_j} & \text{How much bigger (smaller) entity } i \text{ is with respect to entity } j \\ a_{ii}=1 & \text{Every entity has the same size as itself} \\ a_{ji}=\dfrac{1}{a_{ij}} & \text{If entity } i \text{ is } a_{ij} \text{ times bigger (smaller) than entity } j\text{, then entity } j \text{ is } 1/a_{ij} \text{ times smaller (bigger) than entity } i \end{cases} \quad (1)$$

In practice, as Table 1 shows, the judges must estimate only the relative sizes of the upper

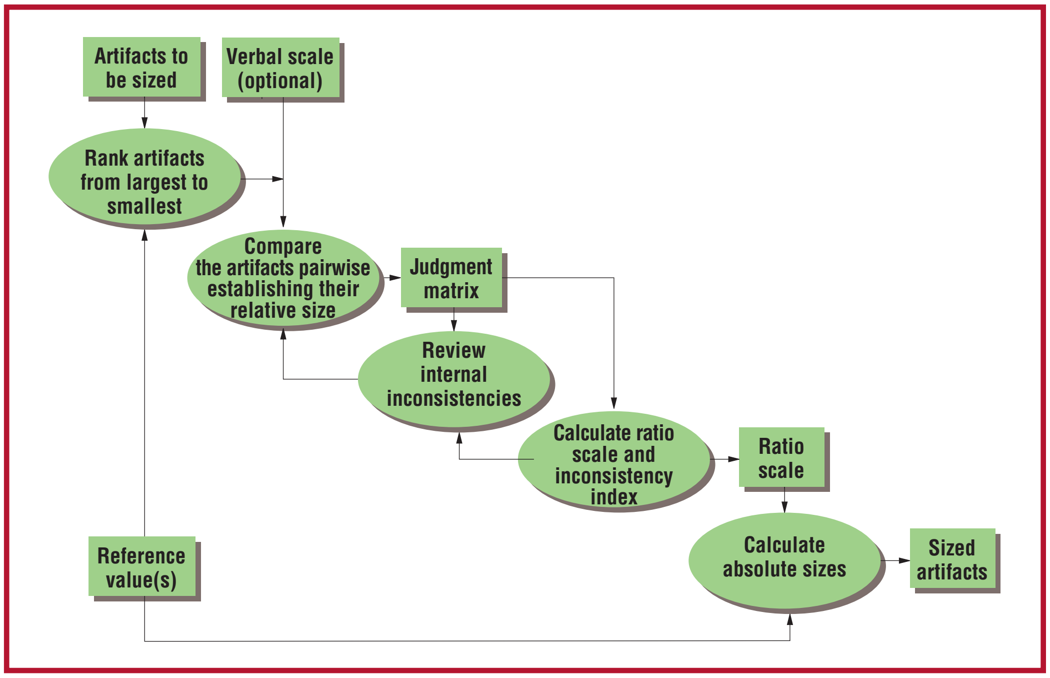

Artifacts to be sized
Verbal scale (optional)
Rank artifacts from largest to smallest
Compare the artifacts pairwise establishing their relative size
Judgment matrix
Review internal inconsistencies
Calculate ratio scale and inconsistency index
Ratio scale
Calculate absolute sizes
Sized artifacts
Reference value(s)

Figure 2. The paired-comparisons estimation process.

diagonal elements of the matrix, because all the other values can be derived from them.

The element "$a_{12} = 4$" in the example in Table 1 expresses the fact that entity D has been judged four times bigger than B. Notice that as shown by the relations D / C = 6, D / A = 7.5, and C / A = 2, the judgments recorded in the matrix do not need to be perfectly consistent. After all, who knows which is the true value? Remember that we are estimating things that have not been built yet.

Although not a mandatory step, arranging the entities in descending order according to their size makes the rest of the process much easier. As Table 1 shows, when we sort the rows of a judgment matrix in descending order, the comparisons flow in one direction only. For example, entity D will be either equal to or larger than any of the other entities against which it is compared; it will never be smaller. Notice also that within a row, the values to the left of any given column are always smaller than or equal to those to the right. While these properties are irrelevant from the mathematical point of view, they diminish the strain put on the judges by the large number of decisions they must make.

The paired-comparisons method requires the existence of at least one reference entity whose size is known, for example, from a previous development project. First, we rank this entity, as we would any other, by comparing it to every other entity to be sized. Later, we use its size to calculate the absolute size of the entities being estimated.

The choice of a reference entity is an important decision. Entities with sizes in either extreme of the scale tend to amplify any bias that might affect the judgments. To minimize this risk, it is better to choose as reference an entity that will divide the population being estimated into halves or to use two or more references instead of one.

Table 1

Judgment Matrix Example

Entities	D	B	C	A
D		4	6	7.5
B			1.5	2
C				2
A				

The verbal scale

Using a verbal scale simplifies and speeds up the estimation process without jeopardizing the accuracy of the results. Although not an essential part of the methodology, having a shared understanding of how small is "smaller" and how big is "bigger" helps the participants reach consensus in the sizing process. A predefined value scale keeps us from wasting time discussing values down to the second decimal when our judgment error is one or two orders of magnitude bigger than that.

Quoting an earlier work from Ernest H. Weber, Thomas L. Saaty proposes using a scale from 1 to 9 and their reciprocals to pass judgment on the entities being evaluated.[7] Table 2 lists the equivalence between verbal expressions and relative sizes.

Suspecting that the values proposed by Saaty could be different for the software domain, I conducted an informal survey among colleagues; 30 people from different countries and from both industry and academia provided input for the scale. The results suggest that the correspondence between size and verbal description in the software domain is closer to the one shown in Table 3 than to Saaty's.

Calculating a ratio scale and an inconsistency index

A *ratio scale* is a vector $[r_1, r_2, \ldots, r_n]$ in which each number r_i is proportional to the size of entity i. An *inconsistency index* is a number that measures how far away our

Table 2

Saaty's Verbal Scale

Definition	Explanation	Relative value	Reciprocal
Equal size	The two entities are roughly the same size.	1	1.00
Slightly bigger (smaller)	Experience or judgment recognizes one entity as being somewhat bigger (smaller).	3	.33
Bigger (smaller)	Experience or judgment recognizes one entity as being definitely bigger (smaller).	5	.20
Much bigger (smaller)	The dominance of one entity over the other is self-evident; very strong difference in size.	7	.14
Extremely bigger (smaller)	The difference between the entities being compared is of an order of magnitude.	9	.11
Intermediate values between adjacent scales	When compromise is needed.	2, 4, 6, 8	.5, .25, .16, .12

Table 3

Verbal Scale for the Software Domain

Definition	Explanation	Relative value	Reciprocal
Equal size	$E_i/E_j \le 1.25$ (0–25%)	1.00	1.00
Slightly bigger (smaller)	$1.25 < E_i/E_j \le 1.75$ (26–75%)	1.25	.80
Bigger (smaller)	$1.75 < E_i/E_j \le 2.275$ (76–275%)	1.75	.57
Much bigger (smaller)	$2.275 < E_i/E_j \le 5.75$ (276–575%)	4.00	.25
Extremely bigger (smaller)	$5.75 < E_i/E_j \le 10$ (576–1000%)	7.50	.13

judgments are from being perfectly consistent. (A *perfectly consistent* judgment matrix is one in which all its elements satisfy the condition $a_{ij} \times a_{jk} = a_{ik}$ for all *i*, *j*, *k*.)

There are several ways to derive ratio scales and inconsistency indexes from paired-comparisons data, among them Saaty's eigenvalues,[7] averaging over normalized columns,[7] and Gordon Crawford and Cindy Williams' geometric mean procedure.[11] Here, I use Crawford and Williams's approach because of its simplicity and good results. I first calculate the geometric mean of the matrix's rows as

$$v_i = \sqrt[n]{\prod_{j=1}^{n} a_{ij}} \ , \qquad (2)$$

then I calculate the ratio scale as

$$r_i = \frac{v_i}{\sum_{l=1}^{n} v_l} \ , \qquad (3)$$

and finally the inconsistency index as

$$\frac{\sqrt{\sum_{i=1}^{n}\sum_{j>i}^{n}\left(\ln a_{ij} - \ln\frac{v_i}{v_j}\right)^2}}{\frac{(n-1)(n-2)}{2}} \ . \qquad (4)$$

Thus, given the ratio scale $[r_1, r_2, \ldots, r_n]$, we can calculate the absolute sizes of the entities being estimated using the expression

$$Size_i = \frac{r_i}{r_{reference}} * Size_{reference} \ . \qquad (5)$$

If more than one reference value is stipulated, the regression line of the references provided can replace the reference size.

A numerical example

Let's look at a complete numerical example using as the departure point the judgments stated in Table 1. First, using the rules for creating a judgment matrix and the relative size of the entities given earlier as examples, we derive values for the matrix:

$$\begin{bmatrix} 1 & 4 & 6 & 7.5 \\ .25 & 1 & 1.5 & 2 \\ .16 & .7 & 1 & 2 \\ .13 & .5 & 1 & 1 \end{bmatrix} .$$

Applying Equation 2, we calculate the vector of the row's geometric means:

$$\begin{bmatrix} 3.6 \\ .93 \\ .68 \\ .42 \end{bmatrix} .$$

We sum the geometric means

$$\sum v_i = 5.7$$

and then normalize the vector just calculated by dividing it by the sum of the means:

$$\begin{bmatrix} .64 \\ .16 \\ .12 \\ .07 \end{bmatrix} .$$

Assuming that entity C is the reference point and its size is 1.7 KLOC, we can calculate the absolute size of the other entities using the relationship in Equation 5:

$$\begin{bmatrix} \frac{.64}{.12} * 1.7 \\ \frac{.16}{.12} * 1.7 \\ \frac{.12}{.12} * 1.7 \\ \frac{.07}{.12} * 1.7 \end{bmatrix} .$$

The absolute sizes—$Size_D$ = 9.07 KSLOC, $Size_B$ = 2.3 KSLOC, and $Size_A$ = 1.06 KSLOC—with an inconsistency index of 3% are the final outputs of the process.

Implementation

Successful implementation of the paired-comparisons method requires the selection of qualified judges and a tool capable of automating the calculations.

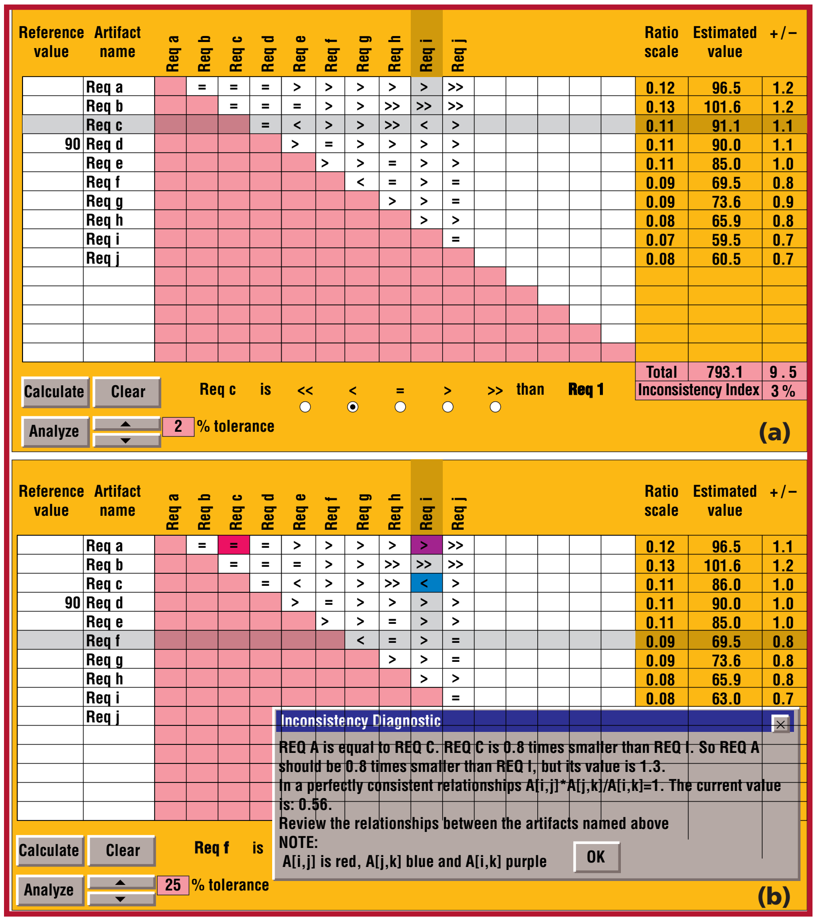
When the number of entities to evaluate is large, you can divide the work among multiple judges. You can also use this approach to minimize the bias introduced by a single judge and to get buy-in to the results. The number of judges used to evaluate *n* entities should not exceed n / 3, otherwise the advantage of the method will be lost because each judge will not get the opportunity to make multiple comparisons for a given entity. A simple way to allocate comparisons to judges is to assign every other comparison to a different judge in a sequential fashion.

At Ericsson, we use the home-grown tool MinimumTime to support the paired-comparisons method. Figure 3a shows MinimumTime's interface, which was designed to reduce the strain put on judges by the large number of comparisons required by the method.

MinimumTime displays all the completed decisions in a matrix structure using a symbolic or numeric format, according to the user's preferences. In keeping with the idea of providing range rather than point estimates, the tool calculates a confidence interval based on the scale dispersion. The tool also provides an analysis capability, shown in Figure 3b, to detect judgment inconsistencies and thus to iteratively refine the initial estimate. The sensibility of this tool can, and should, be adjusted to find only the major discrepancies. Since the true value of the relation is unknown, a certain degree of inconsistency could be considered beneficial.

Software sizing using the paired-comparisons method is especially well suited to the early stages of a development project, when the knowledge available to project team members is mostly qualitative. The mathematics behind the method are foolproof, but the judgments on which the calculations are based are not. For the method to work, those making the comparisons must understand both the functional and the technological dimensions of the things being sized.

Although not conclusive, the results observed so far are promising. Further experimentation is necessary to establish the validity of the verbal scale for the software domain and to verify that the method scales up when used with larger and more complex entities.



Reference value	Artifact name	Req a	Req b	Req c	Req d	Req e	Req f	Req g	Req h	Req i	Req j	Ratio scale	Estimated value	+/−
	Req a		=	=	=	>	>	>	>	>	>>	0.12	96.5	1.2
	Req b			=	=	=	>	>	>>	>>	>>	0.13	101.6	1.2
	Req c				=	<	>	>	>>	<	>	0.11	91.1	1.1
90	Req d					>	=	>	>	>	>	0.11	90.0	1.1
	Req e						>	>	=	>	>	0.11	85.0	1.0
	Req f							<	=	>	=	0.09	69.5	0.8
	Req g								>	>	=	0.09	73.6	0.9
	Req h									>	>	0.08	65.9	0.8
	Req i										=	0.07	59.5	0.7
	Req j											0.08	60.5	0.7
												Total	793.1	9 . 5
												Inconsistency Index		3 %

Calculate Clear Analyze

Req c is << [✓ <] = > >> than Req 1

2 % tolerance

(a)

Reference value	Artifact name	Req a	Req b	Req c	Req d	Req e	Req f	Req g	Req h	Req i	Req j	Ratio scale	Estimated value	+/−
	Req a		=	=	=	>	>	>	>	>	>>	0.12	96.5	1.1
	Req b			=	=	=	>	>	>>	>>	>>	0.13	101.6	1.2
	Req c				=	<	>	>	>>	<	>	0.11	86.0	1.0
90	Req d					>	=	>	>	>	>	0.11	90.0	1.0
	Req e						>	>	=	>	>	0.11	85.0	1.0
	Req f							<	=	>	=	0.09	69.5	0.8
	Req g								>	>	=	0.09	73.6	0.8
	Req h									>	>	0.08	65.9	0.8
	Req i										=	0.08	63.0	0.7
	Req j													

Inconsistency Diagnostic

REQ A is equal to REQ C. REQ C is 0.8 times smaller than REQ I. So REQ A should be 0.8 times smaller than REQ I, but its value is 1.3.
In a perfectly consistent relationships A[i,j]*A[j,k]/A[i,k]=1. The current value is: 0.56.
Review the relationships between the artifacts named above
NOTE:
A[i,j] is red, A[j,k] blue and A[i,k] purple

OK

Calculate Clear Analyze

Req f is

25 % tolerance

(b)

Figure 3. (a) The MinimumTime tool's graphical interface. (b) The consistency analyzer.

Acknowledgments

I thank Tamara Keating and Gaetano Lombardi from Ericsson, Alain Abran from the Université du Québec à Montréal, Norma Chhab-Alperin from Statistics Canada, and Raul Martinez from RMyA for their valuable comments.

References

1. R. Park, *Software Size Measurement: A Framework for Counting Source Statements*, tech. report CMU/SEI-92-TR-20, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1992.
2. A. Albrecht and J. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.*, vol. SE-9, no. 6, 1983, pp. 639–648.
3. *COSMIC—Full Function Points*, Release 2.0, Software Engineering Management Research Lab, Montreal, Sept. 1999; www.lrgl.uqam.ca/cosmic-ffp/manual.html (current 11 Dec. 2000).
4. G. Bozoki, "An Expert Judgment Based Software Sizing Model," Target Software, www.targetsoft-ware.com (current 8 Jan. 2001).
5. J. Karlsson and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software*, vol. 14, no. 5, Sept./Oct. 1997, pp. 67–74.
6. M. Shepperd, S. Barker, and M. Aylett, "The Analytic Hierarchy Processing and Almost Dataless Prediction," *ESCOM-SCOPE '99, Proc. 10th European Software Control and Metrics Conf.*, Shaker Publishing, Maastricht, The Netherlands, 1999.
7. T. Saaty, *Multicriteria Decision Making: The Analytic Hierarchy Process*, RWS Publications, Pittsburgh, 1996.
8. A. Lederer and J. Prasad, "Nine Management Guidelines for Better Cost Estimating," *Comm. ACM*, vol. 35, no. 2, Feb. 1992, pp. 51–59.
9. E. Miranda, "Establishing Software Size Using the Paired Comparisons Method," *Proc. 9th Int'l Workshop Software Measurement*, Université du Québec à Montréal, 1999, pp. 132–142; www.lrgl.uqam.ca/iwsm99/index2.html (current 11 Dec. 2000).
10. E. Miranda, "An Evaluation of the Paired Comparisons Method for Software Sizing," *Proc. 22th Int'l Conf. Software Eng.*, ACM, New York, 2000, pp. 597–604.
11. G. Crawford and C. Williams, *The Analysis of Subjective Judgment Matrices*, tech. report R-2572-1-AF, Rand Corp., Santa Monica, Calif., 1985, pp. xi, 34; www.rand.org/cgi-bin/Abstracts/ordi/getabbydoc.pl?doc=R-2572-1 (current 12 Dec. 2000).

About the Author

Eduardo Miranda is a senior specialist at Ericsson Research Canada and an industrial researcher affiliated with the Research Laboratory in Software Engineering Management at the Université du Québec à Montréal. He is in charge of investigating new management techniques for planning and tracking projects. He received a BS in system analysis from the University of Buenos Aires and an MEng. from the University of Ottawa. He is a member of the IEEE Computer Society and the ACM. Contact him at Ericsson Research Canada, 8400 Decaire Blvd., Town of Mount Royal, Quebec H4P 2N2, Canada; eduardo.miranda@lmc.ericsson.se.