

ESTABLISHING SOFTWARE SIZE USING THE PAIRED COMPARISONS METHOD

Eduardo Miranda

eduardo.miranda@lmc.ericsson.se

Ericsson Research Canada © All rights reserved, August 1999

INTRODUCTION

In our everyday life, we rely on measurement scales and measurement instruments to make all kind of decisions: We listen to the radio to learn about the temperature before deciding how to dress, we look at the Dow Jones index before making an investment, and we calculate the distance separating two cities before starting a trip. But, what do we do when we need to establish the software size at the beginning of a project, before a detailed specification or draft design exists?

Lines of code and function points are good examples of size metrics, but the counting process requires effort and information, which in too many projects, is only available after the project budget and schedule have already been decided. Furthermore, product managers and other stakeholders with non-software background, simply refuse to accept measures that are foreign to them.

To solve the problem of measuring in the absence of a unique and accepted measurement scale, or in cases where a measurement instrument does not exist, the social sciences have adopted the method of paired comparisons for establishing the relative merit of an entity with respect to others. Entities in the social sciences take the form of attitudes, preferences, brand recognition, etc.

The same approach could be used to size software. Although the idea is not new, it has received very little attention in the literature. Earlier attempts include Target Software's Software Sizing Method [1], and more recently a paper by Focal Point AB [2] where an instance of the method, called the Analytic Hierarchy Process, is used to prioritize requirements relative to their cost.

The idea behind the paired comparisons method, is to estimate the size of n entities, be these tasks, requirements, use cases, modules, features or objects, based on their relative largeness as judged by one or more experts.

This article explains the advantages of the approach, the selection of scales, the computational methods and the necessary tool support.

OVERALL APPROACH

Assume that a project requires us to develop three Use Cases¹: **A**, **B**, **C**, whose size in lines of code we want to estimate to use them as input to an estimation model like CoCoMo, and that there is a fourth use case, **D**, whose size is known from a previous development.

Lets suppose that after looking at their functionality and complexity, we judge **Use Case A** to be approximately four times larger than **B**, **B** half the size of **C** and **A** three times bigger than **D**.

From these relationships, captured by Table 1, it is possible to calculate a vector $[r_1, r_2, \dots, r_n]$ called a ratio scale, in which each number r_i is proportional to the size of entity i .

¹ A "use case" is a unit of functionality in the Unified Modeling Language (UML). It encompasses the system response to a given event. Use cases are chosen here, without loss of generality, with the sole purpose of giving concrete meaning to the example. The method is not dependent on their use.

	A	B	C	D
A		4	2	3
B			.5	.75
C				1.5
D				

Element a_{ij} shows the relative size of entity i with respect to entity j . For example, $a_{12} = 4$ express the fact, that Use Case **A** has been judged four times bigger than **B**.

Table 1 – Relative size of each Use Case with respect to the others

Then by using the relationship $\frac{r_i}{r_j} = \frac{Size_i}{Size_j}$ and a reference point, we could calculate the size of each entity.

Table 2 illustrates the procedure given the ratio scale $[0.48, 0.12, 0.24, 0.16]^2$ and **D**'s size³.

Use Case	Largeness	Reference Size	Calculated Size
A	0.48		1,500.0
B	0.12		375.0
C	0.24		750.0
D	0.16	500.0	500.0

Table 2 - Basic calculations

PAIRED COMPARISONS

Why use comparisons instead of an absolute number bestowed by an expert? Why multiple comparisons instead of a simple ranking?

First the human mind is better at establishing differences than at estimating absolute values. Second by comparing each entity to every other entity, the paired comparison method produces redundant values, which compensate for errors and inconsistencies incurred in the judgement process⁴.

By requiring the comparison of each entity against all others, rather than using a single comparison to some vague notion of size buried

in the mind of the estimator, the paired comparisons method forces explicit decisions about the relative size of two entities. These decisions could be later discussed and reviewed by all the stakeholders in the estimation process.

JUDGEMENT MATRICES

A judgement matrix is a matrix of $n \times n$ elements, much like Table 1, where element a_{ij} , express how bigger or smaller, one entity is with respect

to other in terms of the ratio $\frac{size_i}{size_j}$ of the

entities being compared. In more formal terms the elements of a judgement matrix **A** are defined as follows:

² How to calculate this scale will be explained later.

³ Remember this is known from a previous project.

⁴ I have conducted several tests to verify this assumption which are not included here for reasons of space. The results are available on request.

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

In practice all the judges need to do, is provide estimates of the $\frac{size_i}{size_j}$ ratio for the $\frac{n * (n - 1)}{2}$ upper elements of matrix A. All the other values could be derived from them.

If the judge or judges making the comparisons were perfectly consistent in their evaluations, all the elements on the matrix would satisfy the condition $a_{ij}a_{jk} = a_{ik}$ for all i, j, k . In such a case, any column of matrix A would become a ratio scale for the size of the entities being estimated and the process would stop there. However, as judges are seldom so consistent, a mathematical procedure is needed for estimating $[r_1, r_2, \dots, r_n]$. The values listed in the column

labeled "Largeness" in Table 2 above, constitute one of such scales.

There are many ways to calculate ratio scales. In this paper we will discuss only two: Saaty's Eigenvectors Procedure [3] and Crawford & Williams' Geometric Mean Procedure [4]. See Appendix at the end of the article for the detailed calculations.

Once the ratio scale and a measure of the consistency of the judgements are calculated using either approach, the size of the entities is

derived using the $\frac{r_i}{r_j} = \frac{Size_i}{Size_j}$ relationship. Figure

1, shows an step by step estimation according to Saaty's procedure.

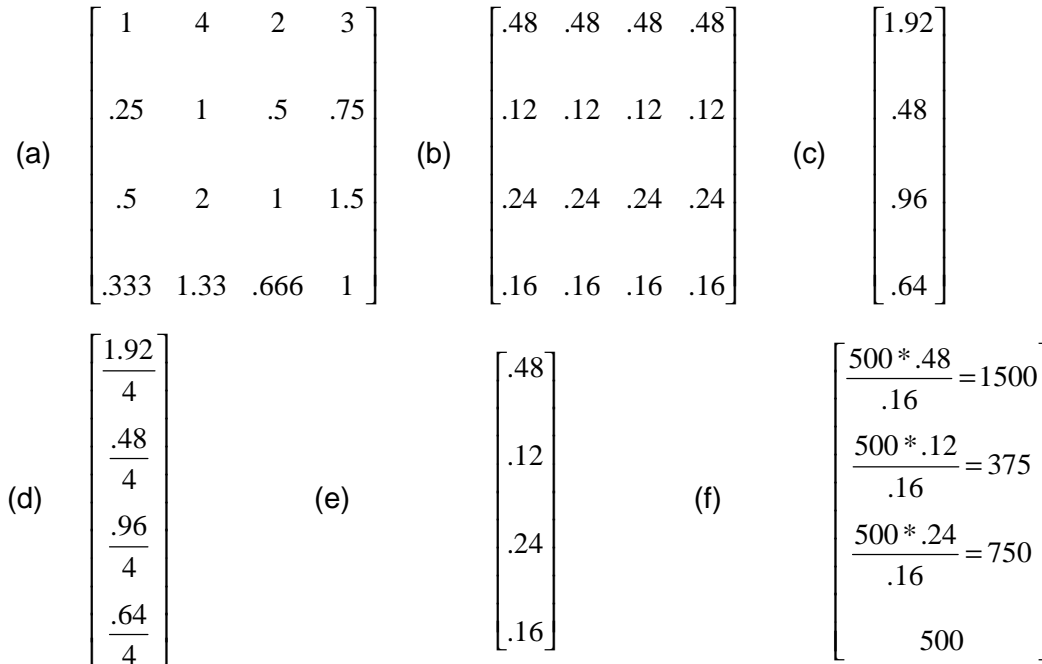


Figure 1 - Numerical solution. (a) Judgement matrix, (b) Normalized matrix, (c) Sum of rows, (d) Normalized sum of rows, (e) Ratio scale, (f) Calculated sizes

The Consistency Ratio (CR)⁵, for the example in Figure 1, is $\frac{0}{0.90} = 0$.

In the preceding example the judge has been perfectly consistent, in reality this is hardly the case, so assuming that the judge evaluating **Use Cases A** through **D** would introduce some inconsistency in the process, we could have gotten a table like Table 3.

	A	B	C	D
A		3.8	2.4	2.6
B			.5	.75
C				1.5
D				

Table 3 - Slightly inconsistent matrix

In this case, the Consistency Ratio would have been $\frac{0.0035}{0.90} = 0.0039$. Saaty considers a consistency ratio of 0.10 or less acceptable. Higher values will require a careful revision of the judgements.

It is important to emphasize, that the estimators do not need to be aware of all these calculations as they could be easily implemented in a spreadsheet.

HOW SMALL IS SMALLER, HOW BIG IS BIGGER?

Although not an essential part of the methodology, having a shared understanding of how small is smaller and how big is bigger, helps reach consensus among participants in the sizing process.

A predefined value scale, keeps us from wasting time discussing values down to the second decimal, when our judgement error is one or two orders of magnitude bigger than that.

The range of sizes exhibited by the entities being compared limits our ability to judge or accurately discriminate size. It is meaningless to compare items whose size is more than one order of magnitude away. A comparative estimation based on an attribute, which can be rated in a scale from 0 to ∞ is likely to be useless.

Quoting an earlier work from Ernest H. Weber, Saaty proposes to use a scale from 1 to 9 and their reciprocals, to pass judgment on the entities been evaluated.

⁵ Refer to appendix at the end of the article for the calculation procedure.

Definition	Explanation	Relative Value	Reciprocal
Equal size	The two entities are roughly the same size.	1	1
Slightly bigger (smaller)	Experience and/or judgement recognize one entity as being somehow bigger (smaller)	3	.33
Bigger (smaller)	Experience and/or judgement recognize one entity as being definitely bigger (smaller)	5	.2
Much Bigger (smaller)	The dominance of one entity over the other is self-evident. Very strong difference in size (smaller)	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 4 - Saaty's verbal scale

An experiment, ran among some colleagues⁶, suggests that the correspondence between size and verbal description in the software domain, is closer to the one provided by Table 5, rather than Saaty's.

The use of the verbal scale simplifies and speeds-up the estimation process without, as will be shown later, jeopardizing the accuracy of the results.

Definition	Explanation	Relative Value	Reciprocal
Equal size	$E_i / E_i \leq 1.25$ (0~25%)	1	1
Slightly bigger (smaller)	$1.25 < E_i / E_i \leq 1.75$ (25 ~ 75%)	1.15	.87
Bigger (smaller)	$1.75 < E_i / E_i \leq 2.275$ (75 ~ 275%)	1.5	.66
Much Bigger (smaller)	$2.275 < E_i / E_i \leq 5.75$ (275 ~ 575%)	3	.33
Extremely bigger (smaller)	$5.75 < E_i / E_i \leq 10$ (575 ~ 1000%)	6	.16

Table 5 - Software verbal scale

⁶ Thirty people from different industries, countries and academia provided their input for this scale. This table might need to be reformulated if the entities being compared are far apart from each other.

1. Actual⁷ E_i values in KSLOC: [50, 30, 25, 24, 24, 10, 9, 8, 5, 5];

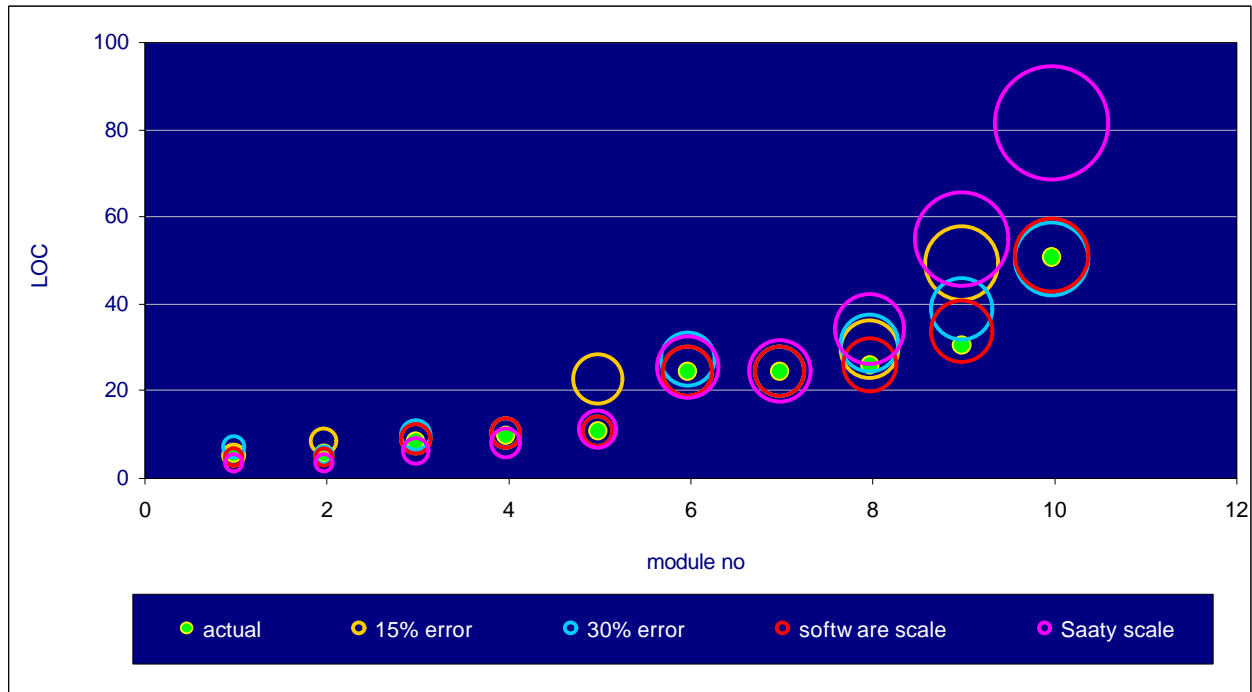


Figure 2 - Results of experiment No. 1

ACCURACY AND PRECISION

In the theory of measurements, accuracy is defined as how close the measurement is with respect to the true value or known input, and precision as the ability to reproduce a set of measurements with a given accuracy. The closer the individual measurements are to one another, the higher the precision.

As with any other measurement instrument, it is important to understand the accuracy and precision that could be expected when using the paired comparison method.

In order to measure the accuracy and precision of the method, two experiments were run with the results illustrated by Figures 2 and 3. In these charts, the encircled area corresponds to the standard deviation of the observations.

The first experiment was based on simulated inputs under the conditions described below.

2. Estimation with judgements $(E_i / E_j) + \mathcal{E}$, where \mathcal{E} is a random variable with normal distribution, $\mu = 0$ and $\sigma = .30 (E_i / E_j)$. In other words, assuming that 68% of the time, the judgements will be within 30% of their true value;
3. Same as 2, but with $\sigma = .15$;
4. Estimation according to the verbal scale for the software domain; and
5. Estimation according to Saaty's verbal scale.

The second experiment was based in a survey of 30 professionals and graduate students, who were asked to assess the absolute or relative size of the data structures⁸ mentioned in the chart, using one of three methods:

⁷ Remember that in a real situation, we do not know these values with the exception of those, one or more, we plan to use as reference.

⁸ Jurors were provided with a brief specification, as assumptions about the language to use, amount of error checking and comments, such as to provide a common framework for the estimation. The actuals are an average of actual programs extracted from libraries written in C,

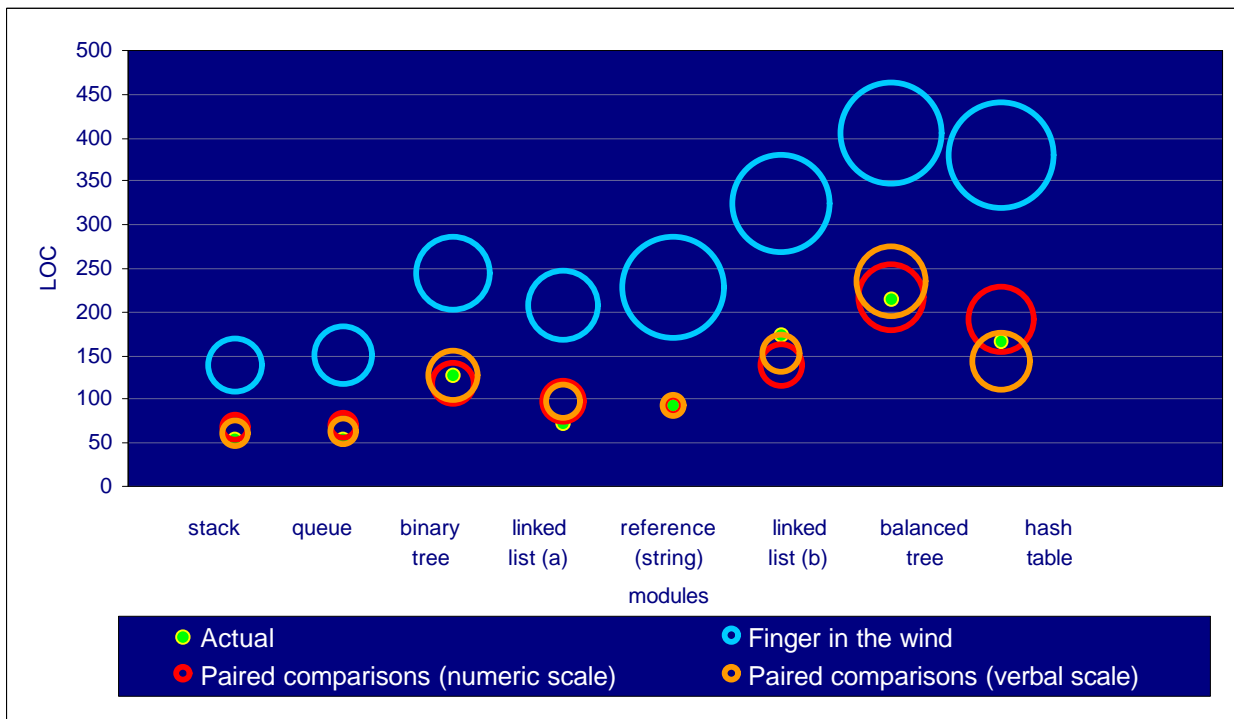


Figure 3 - Results for experiment No. 2

1. **Finger in the wind**, specify the absolute size of the data structures based on our best knowledge;
2. **Paired comparisons numeric scale**, in this case the relative largeness of one data structure with respect to the others was rated using a number; and
3. **Paired comparisons verbal scale**, in this case the relative largeness of one data structure with respect to the others was rated using the verbal scale for the software domain.

As could be easily seen, the paired comparisons method produces accurate and precise results for all the scenarios evaluated.

For the second experiment, it is also possible to conclude with a 95% confidence⁹, that there are significant differences between the “finger in the wind” type of estimation and the results obtained using paired comparisons. Furthermore, using the same kind of test, it is possible to conclude that there are not significant differences between

the results obtained based on the use of either, the numeric or the verbal scales.

Two other interesting observations arising from the experiment are:

- The high variability, of the “finger in the wind approach”, which is almost two to three times bigger than the corresponding to the paired comparisons method; and
- The high correlation, $r = .979$, existing between the relative sizes of the modules independent of the estimation method employed. This seems to corroborate the premise that the human mind is better at establishing differences than at guessing absolute values.

IMPLEMENTATION

The paired comparisons method for software sizing is specially well suited for the early stages of a development project or during feasibility studies, when the knowledge available to the members of the project team, is mostly qualitative.

The mathematics of the method, are fool proof, the judgments on which the calculations are based, are not. In order to produce a valid assessment of the relative largeness of an entity,

Ada and C++ and adjusted for the functionality defined in the questionnaire.

⁹ In order to avoid making assumptions about the underlying distribution of the answers to the questionnaire, a non-parametric *t*-Test for samples with unequal variance was used.

that will divide the population being estimated in halves, or to use two references instead of one. References could be obtained from previous development efforts in the same domain.

Successful implementation of the paired comparisons method, requires a tool capable of automating the calculations and the tracking of judgements made.

TOOL SUPPORT - MINIMUMTIME

MinimumTime, is an internal Ericsson tool, which implements different sizing methods and estimation techniques, among them the paired comparisons method.

When the number of entities to evaluate is large, the work can be divided among multiple judges; this approach could also be used to minimize the bias introduced by a single judge. The maximum number of judges used to evaluate n entities, should not be bigger than $\lfloor n/3 \rfloor$, otherwise the advantage of the method will be lost as each judge will not get the opportunity to make multiple comparisons for a given entity. A simple way to allocate comparisons to judges is by assigning every other comparison to a different judge in a sequential fashion.

Figure 4, shows the input interface which has been designed to reduce the strain and the feeling of being lost, caused by the large number of comparisons required by the method. The tool does this, by constantly displaying the decisions already made in a matrix format. The inputs could be show in symbolic, as well as in numeric formats.

In keeping with the idea of providing range, rather than point estimates, *MinimumTime* calculates a confidence interval based on the scale dispersion.

The analysis capability, illustrated by Figure 5, provides the means for detecting inconsistencies, so that it is possible through an iterative process, to refine the initial estimate.

Choosing a reference value in either extreme of the scale could result on significant under or over estimation of the entities' size depending on the quality of the judgements. To minimize this risk, the best is to choose as reference, an artifact



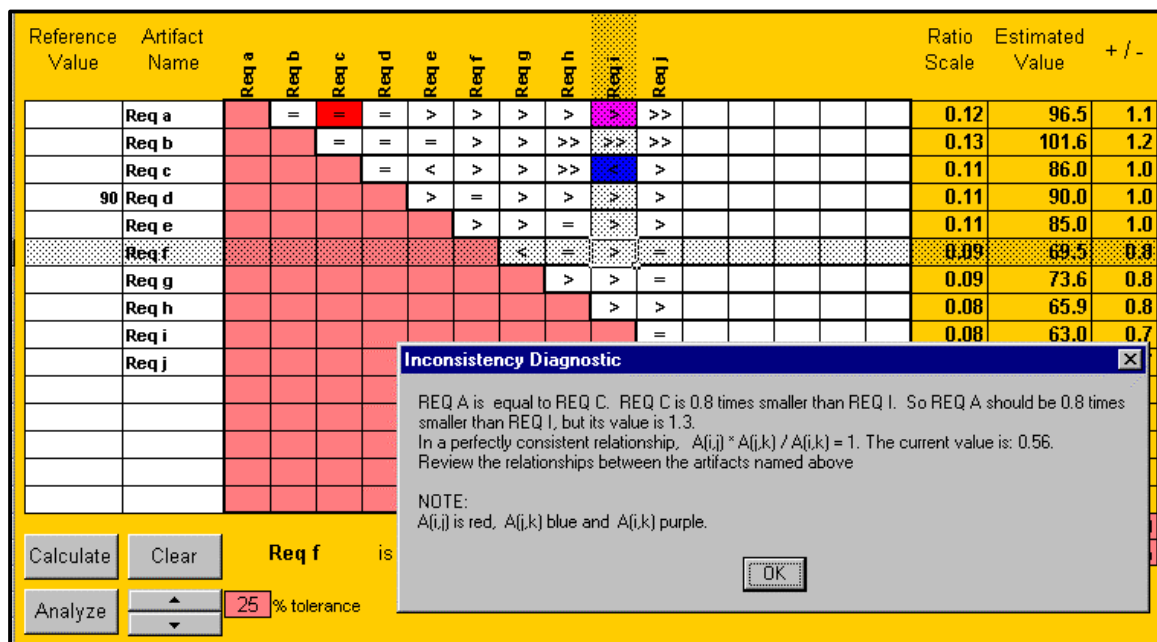


Figure 5 - Consistency analyzer

LOOKING AHEAD

The results observed so far show promise, but 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 or more complex entities.

REFERENCES

1. G. Bozoki, An expert judgement based software sizing model, Lockheed Missiles & Space Company and Target Software
2. J. Karlsson & K. Ryan, A Cost-Value Approach for Prioritizing Requirements, IEEE Software, September/October 1997
3. T. Saaty, Multicriteria Decision Making: The Analytic Hierarchy Process, RWS Publications, 1996
4. G. Crawford and C. Williams, The Analysis of subjective judgement matrices, Rand Corporation, 1985

APPENDIX

SAATY EIGENVECTORS

T. Saaty started investigating the use of judgement matrices on the early '70. He developed multicriteria decision-making procedure called Analytic Hierarchy Process (AHP). From the whole body of AHP knowledge, we'll take only the part corresponding to the derivation of priority scales from subjective judgements. To produce an estimate based on Saaty's proposal, follow the procedure bellow:

- Normalize the columns of the judgement matrix;
- Calculate the sum of the rows;
- Average the sum of the rows; the vector of the averaged sum of rows, is a ratio scale corresponding to the size of the entities $[E_1, E_2, \dots, E_n]$;
- Given a reference size s_k , the expression $s_i = s_k * (r_i / r_k)$ is used to calculate the sizes of entities $[E_1, E_2, \dots, E_k, \dots, E_n]$.

$$\begin{array}{cc}
 \begin{array}{c} \text{(a)} \end{array} \left[\begin{array}{cc} d_{11} = a_{11} / \sum_{i=1}^n a_{i1} & d_{1n} = a_{1n} / \sum_{i=1}^n a_{in} \\ \dots & \dots \\ d_{n1} = a_{n1} / \sum_{i=1}^n a_{i1} & d_{nn} = a_{nn} / \sum_{i=1}^n a_{in} \end{array} \right] & \begin{array}{c} \text{(b)} \end{array} \left[\begin{array}{c} b_1 = \sum_{j=1}^n d_{1j} \\ b_2 = \sum_{j=1}^n d_{2j} \\ \dots \\ b_n = \sum_{j=1}^n d_{nj} \end{array} \right] \\
 \begin{array}{c} \text{(c)} \end{array} \left[\begin{array}{c} r_1 = \frac{b_1}{n} \\ r_2 = \frac{b_2}{n} \\ \dots \\ r_n = \frac{b_n}{n} \end{array} \right] & \begin{array}{c} \text{(d)} \end{array} \left[\begin{array}{c} s_k * \frac{r_1}{r_k} \\ s_k * \frac{r_2}{r_k} \\ \dots \\ s_k \\ \dots \\ s_k * \frac{r_n}{r_k} \end{array} \right]
 \end{array}$$

Figure A1 - Saaty's procedure. (a) Normalized judgement matrix, (b) Sum of rows, (c) Ratio scale, (d) Size calculations

Since judgements are rarely as consistent as in the proposed example, it is important to quantify their inconsistency in order to assess the quality of the results. To the effect, Saaty proposes an index called the *Consistency Ratio* (CR) as a

measure of the degree of agreement among judgements. The CR is calculated by dividing the *Consistency Index* (CI) by the *Random Index* (RI). The RI is a pre-calculated vector, which reflects the random variability of a judgement matrix whose comparisons are ranked from 1 to 9, with the i_{th} element of the vector corresponding to a matrix of order i .

$$(a) \quad CR = \frac{CI}{RI}$$

$$(b) \quad CI = \frac{I - n}{n - 1}$$

$$(c) \quad RI = [0 \quad 0 \quad 0.58 \quad 0.90 \quad 1.12 \quad 1.24 \quad 1.32 \quad 1.41 \quad 1.45 \quad 1.49 \quad 1.51 \quad 1.48 \quad 1.56 \quad 1.57 \quad 1.59]$$

$$(d) \quad I = \frac{\sum_{i=1}^n k_i / r_i}{n}$$

(e)

$$\begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{bmatrix} = \begin{bmatrix} a_{11} & & a_{1n} \\ a_{21} & & \\ \vdots & & \\ a_{n1} & & a_{nn} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} \quad \text{Figure A3}$$

re A2 – (a) Consistency Ratio, (b) Consistency Index, (c) Random Index, (d) & (e) Auxiliary calculations

CRAWFORD & WILLIAMS' GEOMETRIC MEAN PROCEDURE

Back in 1985, Gordon Crawford and Cindy Williams from Rand Corporation published a report for the United States Air Force on the use of judgement matrices for long term planning. They proposed to use the normalized geometric

mean of the row's elements as an estimator of the ratio scale.

(a) Calculate the geometric mean of the rows' element

(b) Normalize the geometric means

(c) Calculate the size of the entities

The inconsistency index (II), is defined by the distance existing between a perfectly consistent and the given judgement matrix. The Crawford & Williams procedure is illustrated by Figure A3.

$$(a) \quad n_i = \sqrt[n]{\prod_{j=1}^n a_{ij}}$$

$$(b) \quad r_i = \frac{n_i}{\sum_{l=1}^n n_l}$$

$$(c) \quad \begin{bmatrix} s_k * \frac{r_1}{r_k} \\ s_k * \frac{r_2}{r_k} \\ \vdots \\ s_k \\ \vdots \\ s_k * \frac{r_n}{r_k} \end{bmatrix}$$

$$(d) \quad II = \frac{\sqrt{\sum_{i=1}^n \sum_{j>i}^n (\ln a_{ij} - \ln c_{ij})^2}}{(n-1)(n-2)} \cdot 2$$

$$(e) \quad c_{ij} = \frac{n_i}{n_j}$$

Figure A3 - Crawford and Williams' procedure. (a) Row's geometric mean, (b) Ratio scale, (c) Size vector, (d) Inconsistency Index, (e) Auxiliary calculation