INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

OPEN ACCESS

**Received:** 01.01.2021

**Accepted:** 26.04.2021

**Published:** 15.06.2021

**Citation:** Tarawneh W (2021) A ComparativeAdoption of COTS Database Components Using the Hybrid AHP — TOPSIS and AHP — VIKOR Methodologies. Indian Journal of Science and Technology 14(21): 1718-1733. https://doi.org/10.17485/IJST/v14i21.4

* **Corresponding author**.

wafa@mutah.edu.jo

**Funding:** None

**Competing Interests:** None



Published By Indian Society for Education and Environment (iSee)

**ISSN**
Print: 0974-6846
Electronic: 0974-5645

# A ComparativeAdoption of COTS Database Components Using the Hybrid AHP — TOPSIS and AHP — VIKOR Methodologies

**Wafa Tarawneh**[1]*

**1** Department of Computer Information System, Faculty of Information Technology, Mu'tah University, Jordan

## Abstract

**Objectives :** This study aims at comparing two recent hybrid methodologies used to evaluate COTS database products based on reusability, including Analytic Hierarchy Process integrated with VIsekriterijumsko KOmpromisno Rangiranje (AHP-VIKOR), and Analytical Hierarchy Process integrated with Order Preference by Similarity to Ideal Solution (AHP-TOPSIS). **Method**: AHP is presented to determine the weight of each evaluation criterion. VIKOR and TOPSIS are used to acquire the final rank of software database alternatives. The evaluation process has been carried out using four database components, namely: Ingress, Oracle 9i, SQL Server 2005 and Microsoft Access. The focus is to compare the difference in the evaluation results between TOPSIS and VIKOR. Both methods are based on an aggregating function that represents closeness to the ideal solution. VIKOR is based on linear normalization whereas TOPSIS used vector normalization to eliminate the units of criterion functions. **Findings**: the solution obtained by TOPSIS method has the shortest distance from the ideal one and farthest from the negative ideal solution. VIKOR method helps to determine a compromise solution that gives a maximum group utility for the majority and minimum for opponents. The results show that the proposed multi-criterion decision-making approach can enhance objectivity the evaluation process of the given database alternatives. In addition, the comparison reveals the effectiveness, and weakness of each method.

**Keywords:** Commercial of the shelf (COTS); Vise Kriterijumska Optimizacija I Kompromisno Resenje (VIKOR); Analytical Hierarchy Process (AHP); Usability; Multiple Criteria Decision Making (MCDM); Normalizing weight; Database; Reuse

## 1 Introduction

Decision-making processes involve a series of steps: identifying the problems, constructing the preferences, evaluating the alternatives, and determining the best alternatives [(1)]. Three kinds of formal analysis can be employed to solve decision-making problems [(2)]: Descriptive analysis is concerned with the problems that Decision Makers (DM) actually solves. Prescriptive analysis considers the methods that DM ought to use to improve their decisions. Normative analysis focuses on the problems that DM should ideally address.

Decision making is extremely intuitive when considering single criterion problems, since we only need to choose the alternative with the highest preference rating. However, when DM evaluate alternatives with multiple criteria, many problems, such as weights of criteria, preference dependence, and conflicts among criteria, seem to complicate the problems and need to be overcome by more sophisticated methods. In order to deal with multiple criteria decision-making (MCDM) problems, the first step is to figure out how many attributes or criteria exist in the problem and how to grasp the way of the problems (i.e., identifying the problems). Next, we need to collect the appropriate data or information in which the preferences of DM can be correctly reflected upon and considered (i.e., constructing the preferences). Further work builds a set of possible alternatives or strategies in order to guarantee that the goal will be reached (i.e., evaluating the alternatives). Through these efforts, the next step is to select an appropriate method to help us to evaluate and outrank or improve the possible alternatives or strategies (i.e., finding and determining the best alternative).

To facilitate systematic research in the field of multiple attribute decision-making, Hwang and Yoon [3] suggested that such problems can be classified into two main categories: multiple attribute decision-making (MADM) and multiple objective decision-making (MODM), based on the different purposes and different data types. The former is applied in the evaluation facet, which is usually associated with a limited number of predetermined alternatives and discrete preference ratings. The latter is especially suitable for the design/planning facet, which aims to achieve the optimal or aspired goals by considering the various interactions within the given constraints. However, conventional MCDM only considers the crisp decision problems and lacks a general paradigm for specific real-world problems, such as group decisions and uncertain preferences. Most MCDM problems in the real world, therefore, should naturally be regarded MCDM problems, which consist of goals, aspects (or dimensions), attributes (or criteria), and possible alternatives (or strategies). More specifically, we can classify MCDM problems in two categories: multiple attribute decision-making (MADM) and multiple objective decision-making (MODM).

The evaluation of alternative commercial off the shelf (COTS) should be considered from various perspectives, for example, database efficiency, database usability, database functionality and so on. The Hybrid Analytical Hierarchy Process-Vlse Kriterijumska Optimizacija I KOmpromisno Resenje in Serbian (AHP-VIKOR), and the Hybrid Analytical Hierarchy Process-Order Preference by Similarity to Ideal Solution (AHP-TOPSIS) described herein are compared and used to rank the alternative COTS components, namely: Oracle 9i, SQL Server 2005 and Microsoft Access . The results prove the effectiveness, illustrate directions for future development, and reveal the strengths and weaknesses of each method.

The remainder of this paper is organized as follows: Section 2 describes the software evaluation methods. Section 3 describes the alternative solutions. Section 4 establishes the evaluation criteria. Section 5 presents the assessment of criteria weight. Section 6 evaluates the alternatives with AHP-VIKOR. Section 7 evaluates the alternatives with AHP-TOPSIS. Section 8 provides the comparison of AHP-TOPSIS and AHP-VIKOR. Finally, the conclusions and future work presented in Section 9.

## 2 Software Evaluation Methods

### The Analytical Hierarchy Process (AHP) Methodology

The AHP is developed by Thomas L. Saaty [4], probably the best-known and most widely used model in decision-making. It is a powerful decision-making tool in determining the priorities among different criteria. The aim of AHP is to identify the optimum alternative and to categorize the others considering the criteria that describe them. The phases involved in AHP methodology are illustrated in Figure 1, below.

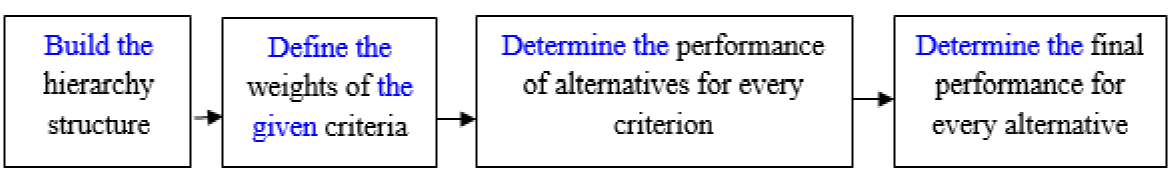


**Fig 1.** Phases of AHP Methodology

The AHP encompasses six basic steps:-

**Step 1:** AHP decomposes a complex decision problem into several sub-problems forming a hierarchy. The goal of the problem is placed a t the top-level, representing the root, and the characteristics are decomposed into several nested sub-levels representing the process of breaking down the criteria into sub-criteria.

**Step 2:** A decision matrix, based on Saaty's nine-point scale, is constructed. The decision maker uses the fundamental 1-9 scale to assess the priority score. In this context, the assessment of 1 indicates equal importance, 3 moderately importance, 5 strongly importance, 7 very strongly importance, and 9 indicate extreme importance (Table 1). The values of 2, 4, 6, and 8 are intermediate values of importance. The decision matrix involves the assessments of each alternative in respect to the decision criteria. If the decision-making problem consists of n criteria and m alternatives; the decision matrix takes the form:

$$\mathrm{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots\cdot & d_m \\ d_{21} & d_{22} & \cdots\cdot & d_{2n} \\ & & & \\ & d_{\mathrm{ml}} & d_{\mathrm{m2}} & \cdots \\ d_m & & & \end{bmatrix}$$

**Step 3:** The third step involves the comparison in pairs of the elements that make up the hierarchy. The aim is to set their relative priorities with respect to each of the elements at the next level up. The Pairwise comparison matrix, based on the Saaty's one-to-nine scale, has the following format, where $w_i$ represents the weight value of the criteria:

$$\underset{\text{Decision-Matrix}}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots\cdot & a_{xm} \end{bmatrix}} = \underset{\text{Pair-Comparison-Matrix}}{\begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \vdots & & & \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots\cdot & \frac{w_n}{w_n} \end{bmatrix}}$$

Assuming n is the number of criteria, then the number of pairwise comparisons between them is equal to n(n-1) / 2. Each value ($a_{ij}$) in theleft-hand-side matrix is matched with the corresponding ($w_i$ / $w_j$) value in the right-hand-side matrix. Each pairwise, $a_{ij} \longleftrightarrow w_i/w_{j,}$ is computed as follows:

$w_i$/ $w_j$= 1 / $a_{ji}$ in all cases except when i = j then $w_i$ / $w_j$ =1. In the comparison matrix, $a_{ij}$ can be interpreted as the degree of preference of ith criteria over jth criteria. It appears that the weight determination of criteria is more reliable when using pairwise comparisons compared to the method of obtaining them directly, because it is easier to make a comparison between two attributes than to make an overall weight assignment.

**Step 4:** Verify the consistency of judgments across the Consistency Index (CI) and the Consistency Ratio (CR)

$$\mathrm{CI} = (\lambda_{\max} - \mathrm{N})/(\mathrm{N} - 1)$$

Where $\lambda_{max}$ is the Eigen value corresponding to the matrix of pair-wise comparisons and n is the number of elements being compared, Consistency ratio (CR) is defined by:

$$\mathrm{CR} = \mathrm{CI}/\mathrm{RCI}$$

where, (RCI) is a random consistency index defined in Table 2. A value of CR less than 0.1 is generally acceptable; otherwise the pair-wise comparisons should be revised to reduce incoherence.

**Step 5 :** The comparison matrix has to be normalized. Therefore, each element has to be divided by the sum of the entries of the corresponding column. In that way, a normalized matrix is obtained in which the sum of all elements vector is 1.

**Step 6:** The eigenvalues of this matrix need to be calculated, which would give the relative weights of criteria. The relative weights obtained in the third step should satisfy the formula: A $_*$ W = $\lambda_{max}$ Where A represents the Pairwise comparison matrix, W represents the weight and $\lambda_{max}$ represents the highest eigenvalues. If there are elements upward on the hierarchy, the weight vector is calculated by multiplying each element (weight coefficient) by its parent at the higher level, this process continues until the top of the hierarchy is reached. The alternative with the highest weight coefficient value should be taken as the best alternative.

**Table 1.** Scale of Relative Importance According to [4]

| Intensity of Importance | Definition | Explanation |
|---|---|---|
| 1 | Equal Importance | Two activities contribute equally to the objective |
| 3 | Moderate importance | Experience and judgment slightly favor one activity |
| 5 | Strong importance | Experience and judgment strongly favor one activity over another |
| 7 | Very strong importance | An activity is favored very strongly over another; it dominance demonstrated in practice |
| 9 | Extreme importance | The evidence favoring one activity over another is of the highest possible order of affirmation |
| 2,4,6,8 | For compromise between the above values | Intermediate values of importance |
| Reciprocal | If variable i has one of the above numbers assigned to it when compared with variable j, then j has the value 1/ number assigned to it when compared with i. More formally if $n_{ij}$ = x then $n_{ji}$ = 1 / x | |

**Table 2.** Average RCI Values

| Consistency ratio index | Number of criteria |
|---|---|
| 0 | 1 |
| 0 | 2 |
| 0.58 | 3 |
| 0.90 | 4 |
| 1.12 | 5 |
| 1.24 | 6 |
| 1.32 | 7 |
| 1.41 | 8 |
| 1.45 | 9 |
| 1.49 | 10 |

## Technique for Order Preference by Similarity to the Ideal Solution (TOPSIS)

TOPSIS, depicted in Figure 2, was initially proposed by Hwang and Yoon [5]. Its basic concept is that the chosen alternative should have the shortest distance from the ideal solution and the farthest from the negative-ideal solution TOPSIS assumes that we have m alternatives (options) and n attributes/criteria and we have the score of each option with respect to each criterion.

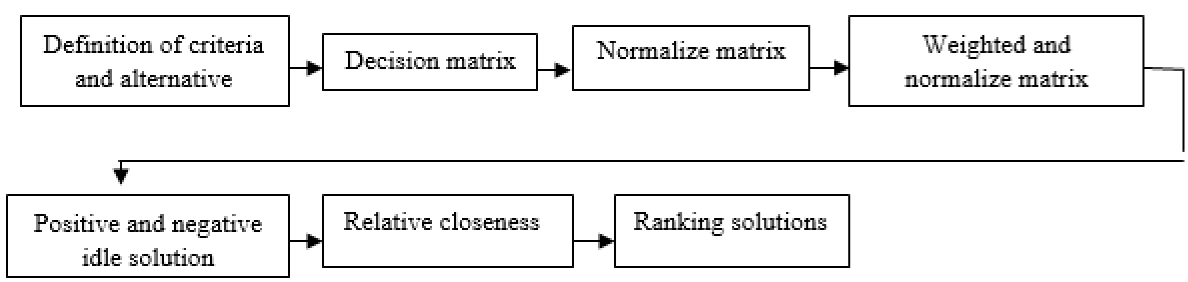


**Fig 2.** TOPSIS Methodology

The steps of TOPSIS model are as follows:

Step 1: Construct normalized decision matrix. This step transforms various attribute dimensions into non-dimensional attributes, which allows comparisons across criteria. Normalize scores or data as follows:

$$r_{ij} = x_{ij} / \left(\Sigma x_{ij}^2\right) \text{ for } i = 1, \ldots, m; j = 1, \ldots, n$$

Step 2: Construct the weighted normalized decision matrix. Assume we have a set of weights for each criteria $w_j$ for j = 1,…n, Multiply each column of the normalized decision matrix by its associated weight, an element of the new matrix is: $v_{ij} = w_j r_{ij}$

Step 3: Determine the ideal and negative ideal solutions. Using the following equations

For ideal solution:

$$A^* = \{v_1{}^* \ldots, v_n^*\}, \text{ where } v_j^* = \left\{\max\left(y_{jj}\right) \text{ if } j \in J; \min\left(v_j\right) \text{ if } j \in J'\right\}$$

For negative solution:

$$A' = \left\{v_1', \ldots, v_n{}'\right\}, \text{ where } v' = \left\{\min\left(v_j\right) \text{ if } j \in J; \max\left(v_{jj}\right) \text{ if } j \in J'\right\}$$

Step 4: Calculate the separation measures for each alternative using the following equations:

$$S_i{}^* = \left[\Sigma\left(v_j^* - v_{jj}\right)^2\right]^{1/2} \quad i = 1, \ldots, m \text{ for the ideal alternatives}$$

$$S_i' = \left[\Sigma\left(v_j' - v_{jj}\right)^2\right]^{1/2} \quad i = 1, \ldots, m \text{ for the negative alternatives}$$

Step 5: Calculate the relative closeness to the ideal solution $C_i{}^*$

$$C_i^* = S_i' / \left(S_i^* + S_i'\right), \quad 0 < C_i^* < 1$$

Step 6: Select the option with $C_i{}^*$ closest to 1

## Vlse Kriterijumska Optimizacija I KOmpromisno Resenje in Serbian (VIKOR)

The VIKOR methodology is depicted in Figure 3 ; it was introduced as an applicable technique to implement within MCDM [6]. It focuses on ranking and selecting from a set of alternatives in the presence of conflicting criteria. The compromise solution, whose foundation was established by Yu [7] and Zeleny [8] is a feasible solution, and it is the closest to the ideal, and here "compromise" means an agreement established by mutual concessions.

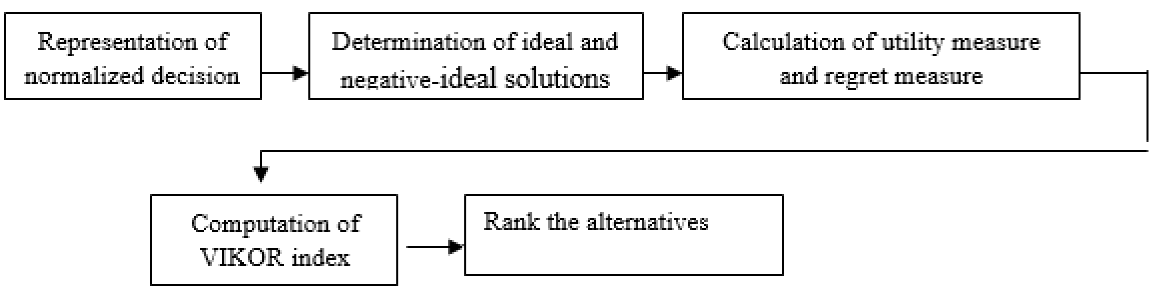


**Fig 3.** VIKOR Methodology

The VIKOR method determines the compromise ranking list and the compromise solution by introducing the multi-criteria ranking index based on the particular measure of "closeness" to the "ideal" solution. The multi-criteria measure for compromise ranking is developed from the Lp-metric used as an aggregating function in a compromise programming method. The levels of regret in VIKOR can be

$$L_{p,i} = \left\{\sum\left[w_i\left(x_j^* - x_{ij}\right) / \left(x_i^* - x_i^-\right)\right]^p\right\}^{1/p} \text{ where } i = 1\ldots n, j = 1\ldots m, \text{ and } 1 \le p < \infty$$

$L_{1,i}$ is defined as the maximum group utility, and $L_{\infty,i}$ is defined as the minimum individual regret of the opponent. The procedure of VIKOR for ranking alternatives can be described in the following steps:

**Step 1:** Determine that best $x_j{}^*$ and the worst $x_j{}^-$ values of all criterion functions, where j = 1, 2,…, n. If the jth criterion represents a benefit then $x_j{}^* = \max f_{ij}$, $f_j{}^- = \min f_{ij}$

**Step 2:** Compute the Si (the maximum group utility) and Ri (the minimum individual regret of the opponent) values, i = 1, 2,…, m by the relations:

$$S_i = L_{1i} = \sum w_i\left(x_i^* - x_{ij}\right) / \left(x_i^* - x_j^-\right) \text{ where } i = 1\ldots m, j = 1\ldots n$$

$R_i = L_{\infty oi} = \max\left[w_i\left(x_i^* - x_{ij}\right) / \left(x_i^* - x_i^-\right)\right]$ where $i = 1\ldots m, j = 1\ldots n$ $w_i$ is the weight of the $j^{th}$ criterion which expresses the relative importance of criteria.

**Step 3:** Compute the value $Q_i$, i = 1, 2, …, m, by the relation

$$Q_i = [v\,(S_i - S^*)\,/\,(S - S^*)] + \left[(1-v)\left[(R_i - R^*)\,/\,\left(R^- - R^*\right)\right]\right]$$

where $S^*$ = min $S_i$, $S^-$ = max $S^*$, $R^*$ = min $R_i$, $R^-$ = max $R^*$ and v is the weight of the strategy of maximum group utility, whereas (1-v) is the weight of the individual regret. Here, when v is larger than 0.5, the index of $Q_i$ follows majority rule [9].

**Step 4:** Rank the alternatives, sorting by the S, R, and Q values in descending order. The results are three ranking lists.

**Step 5**: Propose as a compromise solution the alternative A(1) which is the best ranked by the measure Q (minimum) if the following two conditions are satisfied: C1. "Acceptable Advantage": Q(A(2) – Q(A(1)) >= DQ where: A(2) is the alternative with second position in the ranking list by Q; DQ = 1/(J-1), and J is the number of alternatives. C2. "Acceptable Stability in decision making": The alternative A(1) must also be the best ranked by S or/and R. This compromise solution is stable within a decision-making process, which could be the strategy of maximum group utility (when v > 0.5 is needed), or "by consensus" v about 0.5, or "with veto" v < 0.5). If one of the conditions is not satisfied, then a set of compromise solutions is proposed, which consists of: Alternatives A(1) and A(2) if only the condition C2 is not satisfied, or Alternatives A(1), A(2), ..., A(M) if the condition C1 is not satisfied; A(M) is determined by the relation Q(A(M)) – Q(A(1)) < DQ for maximum M (the positions of these alternatives are "in closeness").

## 3 Proposed Integrated Multi-criteria Decision Methodology

The proposed methodology is designed in such a way that makes the use of Multiple Criteria Decision Making (MCDM) techniques as efficient as possible. Three different techniques, namely AHP, TOPSIS and VIKOR are used; integration of AHP-TOPSIS and integration of AHP- VIKOR are combined in order to rank alternative software according to criteria. The reason for using the well-known AHP technique is to structure the decision hierarchy of the problem. Finally, to rank the alternatives, the most efficient MCDM techniques such as TOPSIS and VIKOR are used. The main steps of the proposed integrated methodology to be elaborated by decision-makers for the database software selection problem are as follows:

Step 1: Define criteria and sub-criteria that are most affecting in the COTS selection problem.

Step 2: Construct a hierarchy decision model for the database software.

Step 3: Find the comparison-matrix for each level (of criteria and sub-criteria) using AHP.

Step 4: Determine the global weight by normalizing the local weight.

Step 5: Use the TOPSIS or VIKOR technique to assess the alternatives.

Step 6: Select the best Database software alternative.

Figure 4 illustrates the process of the proposed integrated methodology.

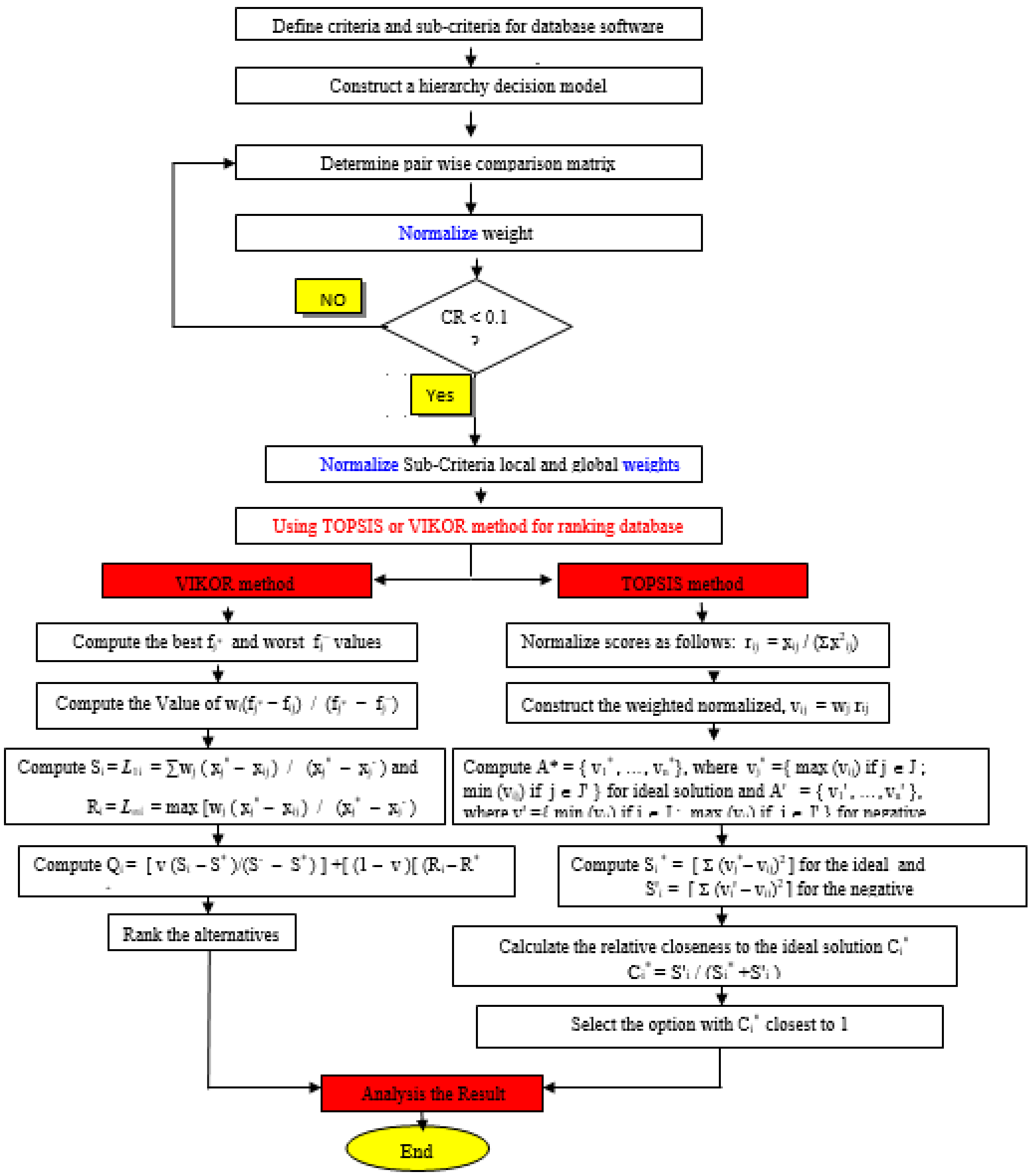


**Fig 4.** Proposed Integrated Methodology to Evaluate and Select COTS software

## 4 Alternative Solutions

The main parameter in defining alternative solutions is the COTS database software; the alternatives are classified into four groups: The Ingres database, Oracle 9i database, SQL Server 2005 database and Microsoft Access database. The 'Usability' of a component should be interpreted as its ability to be used by the application developers and designers when constructing a new software product. The sub-characteristics of 'Usability' are 'Learnability', 'Operability', 'Understandability', and 'Complexity', as described in [9]. 'Learnability' is the capability of the software product to enable the user to learn its application [10], 'Operability' is the capability of the software product to enable the user to operate it and control it [10], 'Understandability' is the capability of the software product to enable the user to understand whether the software is suitable or not, and how it can be used for particular tasks and conditions of uses [10]. 'Complexity' characteristic aims at measuring the complexity of using and integrating the component into the final system.

Based on our work[9], four alternatives of COTS components are considered, and their features are described as follows: (i) Ingres Database is a commercially supported, open-source SQL relational database management system intended to support large commercial and government applications. Ingres Database is fully open source with a global community of contributors. However, Actian Corporation controls the development of Ingres and makes certified binaries available for download, as well as providing worldwide support[11].

Ingres began as a research project at the University of California, Berkeley, starting in the early 1970s and ending in 1985[12]. The original code, like that from other projects at Berkeley, was available at minimal cost under a version of the BSD license. Ingres spawned a number of commercial database applications, including Sybase, Microsoft SQL Server, NonStop SQL and a number of others. Postgres (Post Ingres), a project which started in the mid-1980s, later evolved into PostgreSQL. It is ACID compatible and is fully transactional (including all DDL statements) and is part of the Lisog open-source stack initiative. (ii) Oracle 9i Database (commonly referred to as Oracle RDBMS or simply as Oracle) is an object-relational database management system[3] produced and marketed by Oracle Corporation[13]. Larry Ellison and his two friends and former co-workers, Bob Miner and Ed Oates, started a consultancy called Software Development Laboratories (SDL) in 1977.

SDL developed the original version of the Oracle software. The name Oracle comes from the code-name of a CIA-funded project Ellison had worked on while formerly employed by Ampex[14]. The Oracle 9i Database Components include: Database Management System, Internet Application Server 9i, Report Builder, Java Database Connection, Application Program Interface, COTS Product Crystal Report, and COTS product Web Portal. (iii) SQL Server 2005 is a Microsoft product used to manage and store information. Technically, SQL Server is a "relational database management system" (RDMS). Broken apart, this term means two things. First, that data stored inside SQL Server will be housed in a "relational database", and second, that SQL Server is an entire "management system", not just a database. SQL itself stands for Structured Query Language. This is the language used to manage and administer the database server[15].

## 5 Establishing the evaluation criteria

The database software selection decision is very important in long-term planning for any business. The contribution suggests an evaluation process that serves the purpose of choosing the appropriate COTS component, for example, database software in an organization selected by a group of developers. The evaluation process provides the knowledge that is necessary to confirm the choice of a particular method, and without such knowledge the uncertainty will compromise the benefits. Thus, choosing the appropriate COTS achieves a high degree of reusability and the desired benefits. The starting point for this work is the Rawashdeh and Matalkah model[9] simply because it includes the common software quality characteristics.

As described in[9], the suggested framework is useful for its integrated approach to quality. Each high-level characteristic of database software product is associated with a set of sub-characteristics. A sub-characteristic is, further, represented by sets of software quality attributes. This chain of software quality attributes can be classified into a hierarchy of three levels as shown in Figure 5. At the top level the so-called 'characteristic' from a customer or stakeholders' perspectives, such as 'Usability'. At the second level, the so-called 'Sub-characteristics' or quality factors from a customer or stakeholders' perspectives, such as, 'Learnability', 'Operability', 'Understandability' and 'Complexity'. At the third level are the quality criteria (attributes), which represent technical concepts. At the fourth level, the 'Metric' that measure the quality criteria (attributes) of database software product.

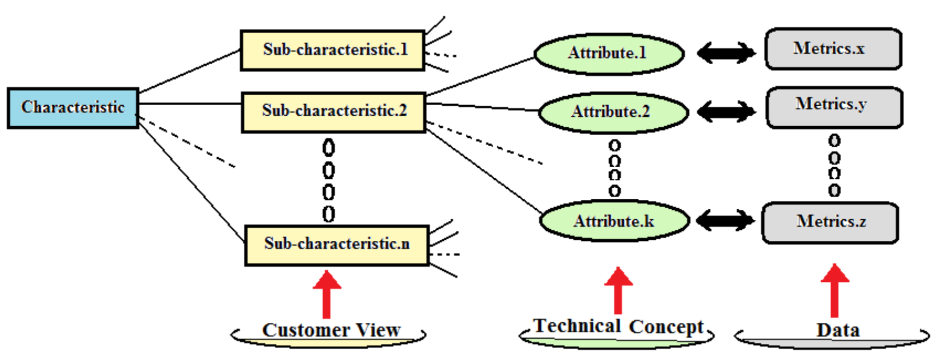


**Fig 5.** Framework of COTS Quality Attributes

The following is the evaluation discussion of the high-level of characteristic 'Usability', along with their associated sub-characteristics. 'Usability' is the capability of the software product to be understood, learned, used and be attractive to the user, when used under specified conditions. 'Usability' is related to the set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. In addition, 'Usability' is the effort required to learn, operate, prepare input, and interpret output of a program [16]. In COTS, most stakeholders of components are the application developers, and designers that have to build applications with them, and end-users that interact with COTS. Thus, the Usability of a component should be interpreted as its ability to be used by the application developer and designer when constructing a new software product. The sub-characteristics of 'Usability' are 'Learnability', 'Operability', 'Understandability', and 'Complexity', as described in [9].

'Learnability': Is the capability of the software product to enable the user to learn its application [10], it requires attention to the needs of the novice and uninitiated users, the uninitiated user is one that has no previous experience with the software or similar software, the novice user has either had some experience with similar software or has limited experience with the software. There is a set of attributes that try to measure the time needed to master some specific task (such as usage or configuration). Here in, 'Learnability' attributes will be decomposed into the following:

1. Time to Use: Attribute measures the average time needed for a developer to learn how to correctly use the database software component,
2. Time to Configure: Attribute measures the average time needed for a developer to learn how to correctly configure the component, and for properly understanding its configuration parameters.

'Understandability': Is the capability of the software product to enable the user to understand whether the software is suitable or not, and how it can be used for tasks and conditions of uses [10]. This attribute deals with the component documentation, demos, and tutorials available. It is important to notice that this characteristic is closely related to 'Learnability', since for an entity or service to be learned; it has to be understood first. Thus, under these characteristics we have grouped those attributes that facilitate the 'Understandability' of a component, and therefore influence its 'Learnability'. Here in, the 'Understandability' attributes will be decomposed into the following:

1. Documentation: consists of end-user documentation. Attribute measures the quality of the user documentation, in terms of its completeness, clarity, and usefulness. Computer Documentations. Attributes, specifies whether the components provide any kind of documentation that can be used by component tools for understanding its services (e.g. User Manual, ERM or DFD).
2. Training that indicates whether training courses are available for the software component,
3. Support measures the level of support provided by the vendor through surveys, web, discussion, groups, interview, and news.

'Operability' Is the capability of the software product to enable the user to operate and control it [10], or the ease of operating a program. Here in, the 'Operability' attributes will be decomposed into the following:

1. Effort for Operating attribute indicate the level of effort needed to properly operate the software component.
2. Administrability attribute indicates the level of effort needed to properly administer the software component.

'Complexity': This characteristic aims at measuring the complexity of using and integrating the component into the final system. For that we will measure the number of provided and required interface, and average number of operations per interface. Here in, the 'Complexity' attributes will be decomposed into the 'Required Interface': number of interfaces that the COTS component requires from other components to operate.

A new framework, dedicated to COTS-based reuse, has been built to support a standard set of software quality characteristics suitable for evaluating COTS components, along with newly defined sets of sub-characteristics associated with them. The new framework avoids some of the limitations found in another existing framework. The new framework ignores quality characteristics that are not applicable to COTS components and is empowered with new ones that are. The same new framework has been further enhanced through identifying new attributes for the quality sub-characteristics in the framework and defining metrics rules to measure the quality of these new attributes. Figure 6 shows the breakdown of the attributes along with their associated metrics and criteria.

In this contribution, the framework is tested with Integrated AHP-VIKOR, and AHP-TOPSIS Methodologies to evaluate and select the favorable COTS database product among Ingres, Oracle 9i, SQL Server 2005 and Microsoft Access. The comparison between AHP-TOPSIS and AHP-VIKOR, accordingly, to evaluate and select the favorable COTS database product among Ingress, Oracle 9i, SQL Server 2005 and Microsoft Access.

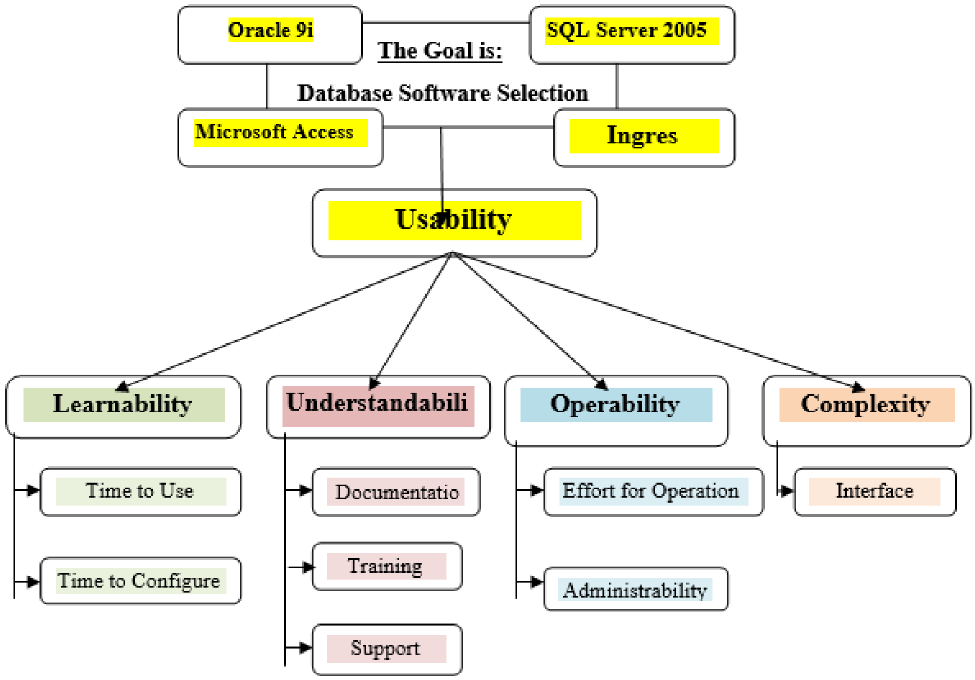


**Fig 6.** Characteristics and Sub-characteristics of Usability

## 6 Assessment of criteria weight

A new framework, dedicated to COTS-based reuse, has been built to support a standard set of software quality characteristics suitable for evaluating COTS components, along with newly defined sets of sub-characteristics associated with them. The new framework avoids some of the limitations found in another existing framework. The new framework ignores quality characteristics that are not applicable to COTS components and is empowered with new ones that are. The same new framework has been further enhanced through identifying new attributes for the quality sub-characteristics in the framework and defining metrics rules to measure the quality of these new attributes. Figure 6 shows the breakdown of the attributes along with their associated metrics and criteria.

In this contribution, the framework is tested with Integrated AHP-TOPSIS and AHP-VIKOR Methodologies to evaluate and select the favorable COTS database product among Ingres, Oracle 9i, SQL Server 2005 and Microsoft Access. The final results of the computation process on the four alternatives are compared among each other.

Using Saaty scaling-table, and the AHP six steps, a weight value is assigned for each of the characteristics, namely: 'Learnability', 'Understandability', 'Operability', and 'Complexity'. The outcome is shown in Matrix-1, below.

**Matrix-1: Pairwise Comparisons Judgment for the Sub-Characteristics According to 'Usability'**

| Usability | Learnability | Understandability | Operability | Complexity | Priority |
|---|---|---|---|---|---|
| Learnability | 1.0 | 2.0 | 5.0 | 5.6 | 0.52 |
| Understandability | 0.50 | 1.0 | 2.50 | 2.80 | 0.27 |
| Operability | 0.20 | 0.40 | 1.0 | 1.12 | 0.11 |
| Complexity | 0.18 | 0.30 | 0.89 | 1.0 | 0.10 |
| CR = 0.013 | | | | ∑ Priority = 1.0 | |

With regard to 'Learnability', a weight value is assigned for each of the sub-characteristics: 'Time to Use', 'Time to Configure'. The outcome is shown in Matrix-2, below.

**Matrix-2: Pairwise Comparisons Judgment for the Sub-Characteristics According to**

| Learnability | Time to Use | Time to Configure | Priority |
|---|---|---|---|
| Time to Use | 1.0 | 2.0 | 0.67 |
| Time to Configure | 0.5 | 1.0 | 0.33 |
| CR = 0.007 | ∑ Priority = 1.0 | | |

With regard to 'Understandability' a weight value is assigned for each of the sub-characteristics: 'Documentation', 'Training', and 'Support'. The outcome is shown in Matrix-3, below.

**Matrix-3: Pairwise Comparisons Judgment for the Sub-Characteristics According to 'Documentation', 'Training', and 'Support'**

| Understandability | Documentation | Training | Support | Priority |
|---|---|---|---|---|
| Documentation | 1.0 | 2.0 | 2.0 | 0.50 |
| Training | 0.50 | 1.0 | 1.0 | 0.25 |
| Support | 0.50 | 1.0 | 1.0 | 0.25 |
| CR = 0.0 | ∑ Priority = 1.0 | | | |

With regard to 'Operability' a weight value is assigned for each of the sub-characteristics: 'Effort for Operating' and 'Administrability'. The outcome is shown in Matrix-4, below.

**Matrix-4: Pairwise Comparisons Judgment for the Sub-Characteristics According to 'Effort for Operating' and 'Administrability'**

| Operability | Effort for Operating | Administrability | Priority |
|---|---|---|---|
| Effort for Operating | 10. | 0.5 | 0.33 |
| Administrability | 2.0 | 1.0 | 0.67 |
| CR = 0.007 | ∑ Priority = 1.0 | | |

With regard to 'Complexity' a weight value one is assigned to the attribute 'Required Interface' because the 'Complexity' sub-characteristic is decomposed to only one attribute.

## 7 Evaluate alternatives with AHP-VIKOR Approach

The VIKOR method is applied in order to rank the alternative database software. In order to demonstrate this process in the current methodology the four database components, Ingres, Oracle 9i, SQL Server 2005, and Microsoft Access are used. The first step, the global weights of each Criteria and Sub-criterion, including: Learnability, Understandability, Operability, Complexity Time to use, Time to configure, Documentation, Training, Support, Effort for operating, Administrability and Required interface are calculated by AHP as shown in Table 3, and thus can be used as the input to the VIKOR method. Therefore, by using the scale in Table 1, the decision-makers are asked to evaluate the alternatives according to each sub-criterion, as illustrated in Table 4, below.

**Table 3.** The Normalized Sub-Criteria Weightings

| Criteria | Weight | Sub-Criteria | Weight | Level Two |
|---|---|---|---|---|
| Learnability | 0.52 | Time to Use | 0.67 | 0.3484 |
| | | Time to Configure | 0.33 | 0.1716 |
| Understandability | 0.27 | Documentation | 0.50 | 0.135 |
| | | Training | 0.25 | 0.0675 |
| | | Support | 0.25 | 0.0675 |
| Operability | 0.11 | Effort for Operating | 0.33 | 0.0363 |
| | | Administrability | 0.67 | 0.0737 |
| Complexity | 0.10 | Required Interface | 1.0 | 0.10 |
| | Weight =1.0 | | | Level Two =1.0 |

**Table 4.** Input Values of the VIKOR Analysis

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 5 | 5 | 6 | 7 | 4 | 4 | 6 | 5 |
| Oracle 9i | 7 | 7 | 6 | 5 | 8 | 5 | 6 | 6 |
| SQL Server | 3 | 5 | 4 | 5 | 3 | 4 | 5 | 4 |
| MS-Access | 8 | 6 | 6 | 8 | 9 | 6 | 5 | 7 |
| Weight | 0.3484 | 0.1716 | 0.135 | 0.0675 | 0.0675 | 0.0363 | 0.0737 | 0.10 |

The second step is to calculate $f_j^*$ and $f_j^-$ associated with Ingres Oracle 9i, SQL Server and Microsoft Access. The outcomes are illustrated in Table 5, below.

**Table 5.** Best $f_j^*$ and Worst $f_j^-$ Values

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| $f_j^*$ | 8 | 7 | 6 | 8 | 9 | 6 | 6 | 7 |
| $f_j^-$ | 3 | 5 | 4 | 5 | 3 | 4 | 5 | 4 |
| Weight | 0.3484 | 0.1716 | 0.135 | 0.0675 | 0.0675 | 0.0363 | 0.0737 | 0.10 |

The third step is to calculate the value of $w_i(f_j^* - f_{ij}) / (f_j^* - f_j^-)$, as in Table 6 , below.

**Table 6.** Value of $\mathbf{w_i}(f_j^* - f_{ij}) / (\mathbf{f_j}^* - \mathbf{f_j}^-)$

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 0.2090 | 0.1716 | 0.0000 | 0.0225 | 0.0562 | 0.0363 | 0.0000 | 0.0667 |
| Oracle 9i | 0.0697 | 0.0 | 0.0000 | 0.0675 | 0.0113 | 0.0182 | 0.0000 | 0.0333 |
| SQL Server | 0.3484 | 0.1716 | 0.135 | 0.0675 | 0.0675 | 0.0363 | 0.0737 | 0.1000 |
| MS-Access | 0.0000 | 0.0858 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0737 | 0.0000 |

The fourth step is to calculate $S_i$ and $R_i$ using the following relations. The outcomes are illustrated in Table 7, below.

$$S_i = L_{1i} = \sum w_j \left(x_i^* - x_{ij}\right) / \left(x_i^* - x_i^-\right) \text{ and}$$
$$R_j = L_{\infty c} = \max \left[w_i \left(x_i^* - x_{ij}\right) / \left(x_i^* - x_j^-\right)\right]$$

**Table 7.** The Value of $\mathbf{S_i}$ and $R_i$

| | $S_i$ | $R_i$ |
|---|---|---|
| Ingres | 0.5623 | 0.0697 |
| Oracle 9i | 0.2000 | 0.116 |
| SQL Server 2005 | 1.0000 | 0.3484 |
| MS-Access | 0.1595 | 0.0858 |

The fifth step is to calculate the $Q_i$ using the following relation with value v = 0.5. The outcomes are illustrated on Table 8, below.

$$Q_i = [v(S_i - S^*)/(S^- - S^*)] + [(1-v)[(R_i - R^*)/(R^- - R^*)]]$$
$$\text{Where } S^* = 0.1595,\ S^- = 1.0, R^* = 0.0697,\ \text{and } R^- = 0.3484\,,$$

**Table 8.** The Corresponding Values of $Q_i$

| | $S_i$ | $R_i$ | $Q_i$ |
|---|---|---|---|
| Ingres | 0.7017 | 0.3484 | 0.823 |
| Oracle 9i | 0.2430 | 0.116 | 0.1328 |
| SQL Server 2005 | 0.1595 | 0.0858 | 0.0289 |
| MS-Access | 1.0 | 0.3484 | 1.0000 |

The sixth step rank the alternatives, by sorting S, R and Q values, in descending order as illustrated on Table 9, below.

**Table 9.** Sorting S, R and Q Values, in Descending Order

| | $S_i$ | $R_i$ | $Q_i$ |
|---|---|---|---|
| SQL Server 2005 | 0.1595 | 0.0858 | 0.0289 |
| Oracle 9i | 0.2430 | 0.116 | 0.1328 |
| Ingres | 0.7017 | 0.3484 | 0.8230 |
| MS-Access | 1.0000 | 0.3484 | 1.0000 |
| Compromise solutions | SQL Server 2005 | SQL Server 2005 | SQL Server 2005 |

The alternative with first position is SQL Server 2005 with Q value = 0.0289, and Oracle 9i is the alternative with second position with Q value = 0.1328. As DQ = 1/(J-1) = 1/(4-1)= 0.333, so Q(Oracle 9i ) – Q(SQL Server 2005) = 0.1328 – 0.0289 = 0.1039 < 0.333. Which does not satisfy the first condition one (C1) in step 5 Q(SQL Server 2005) – Q(Oracle 9i ) >= 0.333, but the alternative SQL Server 2005 is the best ranked bycondition (C2) in step 5 substituting the corresponding values, we get:

Q(Oracle 9i ) – Q(SQL Server 2005) = 0.1328 – 0.0289 = 0.1039 < 0.333,

Q(Ingres) – Q(SQL Server 2005) = 0.8230 – 0.0289 = 0.7941 > 0.333

Therefore, SQL Server 2005 and Oracle 9i are both compromise solutions.

## 8 Evaluating Alternatives with AHP-TOPSIS Approach

The TOPSIS method is applied in order to rank the alternative database software in the same way the VIKOR was applied in the previous section. In order to demonstrate this process in the current methodology the four database components, Ingres, Oracle 9i, SQL Server 2005, and Microsoft Access are used. The first step, the global weights of each Criteria and sub-criterion' Learnability, Understandability, Operability, Complexity Time to use, Time to configure, Documentation, Training, Support, Effort for operating, Administrability and Required interface are calculated by AHP as shown in Table 3, and thus can be used as the input to the TOPSIS method. Therefore, by using the scale in Table 1, the decision-makers are asked to evaluate the alternatives according to each sub-criterion, as illustrated in Table 10, below.

**Table 10.** Input Values of the TOPSIS Analysis

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 5 | 5 | 6 | 7 | 4 | 4 | 6 | 5 |
| Oracle 9i | 7 | 7 | 6 | 5 | 8 | 5 | 6 | 6 |
| SQL Server | 3 | 5 | 4 | 5 | 3 | 4 | 5 | 4 |
| MS-Access | 8 | 6 | 6 | 8 | 9 | 6 | 5 | 7 |
| Weight | 0.3484 | 0.1716 | 0.135 | 0.0675 | 0.0675 | 0.0363 | 0.0737 | 0.10 |

The second step is to calculate $\left(\sum x^2 \text{ ij}\right)^{1/2}$ for each column as illustrated in Table 11, below.

**Table 11.** Calculating $\left(\sum x^2 \text{ ij }\right)^{1/2}$ for Each Column

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 25 | 25 | 36 | 49 | 16 | 16 | 36 | 25 |
| Oracle 9i | 49 | 49 | 36 | 25 | 64 | 25 | 36 | 36 |
| SQL Server | 9 | 25 | 16 | 25 | 9 | 16 | 25 | 16 |
| MS-Access | 64 | 36 | 36 | 64 | 81 | 36 | 25 | 49 |
| $\sum X_{ij}{}^2$ | 147 | 135 | 124 | 163 | 170 | 93 | 122 | 126 |
| $(\sum X_{ij}{}^2)^{0.5}$ | 12.124 | 11.619 | 11.136 | 12.767 | 13.038 | 9.644 | 11.045 | 11.223 |

The third step is to divide each column by $\left(\sum x^2 \text{ ij }\right)^{1/2}$ to obtain $r_{ij}$ as illustrated in Table 12, below.

**Table 12.** Dividing Each Column by $\left(\sum x^2 \text{ ij }\right)^{1/2}$ to Obtain $\mathbf{r}_{ij}$

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 0.4124 | 0.4303 | 0.5388 | 0.5483 | 0.3068 | 0.4148 | 0.5432 | 0.4455 |
| Oracle 9i | 0.5774 | 0.6025 | 0.5388 | 0.3916 | 0.6136 | 0.5185 | 0.5432 | 0.5346 |
| SQL Server | 0.2474 | 0.4303 | 0.3592 | 0.3916 | 0.2300 | 0.4148 | 0.4527 | 0.3564 |
| Microsoft Access | 0.6600 | 0.5164 | 0.5388 | 0.6266 | 0.6903 | 0.6221 | 0.4527 | 0.6237 |

The fourth step multiply each column by $w_j$ to get $v_{ij}$. as illustrated in Table 13. The fifth step is to determine ideal solution $A^* = \{v_1^*, \ldots, V_n^*\}$ where $v_j^* = \{\max(v_{ij}) \text{ if } j \in J; \min(v_{ij}) \text{ if } j \in J'\}$ so A* = {0.2299, 0.1034, 0.0727, 0.0423, 0.0466, 0.0226, 0.0400, 0.0624}.

**Table 13.** Multiplying Each Column by $w_j$ to Obtain $v_{ij}$

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface |
|---|---|---|---|---|---|---|---|---|
| Ingres | 0.1437 | 0.0738 | 0.0727 | 0.0370 | 0.0207 | 0.0151 | 0.0400 | 0.0446 |
| Oracle 9i | 0.2013 | 0.1034 | 0.0727 | 0.0264 | 0.0414 | 0.0188 | 0.0400 | 0.0535 |
| SQL Server | 0.0862 | 0.0738 | 0.0485 | 0.0264 | 0.0155 | 0.0151 | 0.0334 | 0.0356 |
| Microsoft Access | 0.2299 | 0.08886 | 0.0727 | 0.0423 | 0.0466 | 0.0226 | 0.0334 | 0.0624 |

The sixth step is to find the negative ideal solution A' = { $v_1$', …,$v_n$' }, where $y' = \{\min(v_{ij}) \text{ if } j \in J; \max(v_{ij}) \text{ if } j \in J'\}$ so A' = {0.0862, 0.0738, 0.0485, 0.0264, 0.0155, 0.0151, 0.0334, 0.0356 }. The seventh step is to determine separation from ideal solution:

$S_i^* = \left[\sum\left(v_j^* - v_{ij}\right)^2\right]^{1/2}$ for each row, as illustrated in Table 14, below.

**Table 14.** The Separation from Ideal Solution

| | Time to Use | Time to Configure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface | $S_i^*$ |
|---|---|---|---|---|---|---|---|---|---|
| Ingres | 0.0074 | 0.0009 | 0.0 | 0.0001 | 0.0007 | 0.0001 | 0.0 | 0.0003 | 0.097 |
| Oracle 9i | 0.0008 | 0.0 | 0.0 | 0.0003 | 0.0001 | 0.0001 | 0.0 | 0.0001 | 0.037 |
| SQL Server | 0.0206 | 0.0009 | 0.0006 | 0.0003 | 0.0010 | 0.0001 | 0.0001 | 0.0001 | 0.154 |
| MS-Access | 0.0 | 0. 0002 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0001 | 0.0 | 0.017 |

The eighth step is to find the separation from negative ideal solution:

$S_i' = \left[\sum\left(v_j' - v_{ij}\right)^2\right]^{1/2}$ for each row as illustrated in Table 15

**Table 15.** The Separation from Negative Ideal Solution

| | Time to Use | Time to Config-ure | Documentation | Training | Support | Effort for Operating | Administrability | Required Interface | $S_i$’ |
|---|---|---|---|---|---|---|---|---|---|
| Ingres | 0.0033 | 0.00 | 0.0006 | 0.0001 | 0.0001 | 0.0 | 0.0001 | 0.0001 | 0.066 |
| Oracle 9i | 0.0132 | 0.0009 | 0.0006 | 0.0 | 0.0001 | 0.0001 | 0.0001 | 0.0003 | 0.124 |
| SQL Server | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| MS-Access | 0.0206 | 0.0002 | 0.0006 | 0.0003 | 0.0010 | 0.0001 | 0.0 | 0.0007 | 0.153 |

**Table 16.** The Relative Closeness to the Ideal Solution

| | $S_i^*$ | $S_i$’ | | $C_i^*$ |
|---|---|---|---|---|
| Ingres | 0.097 | 0.066 | 0.163 | 0.405 |
| Oracle 9i | 0.037 | 0.124 | 0.161 | 0.77 |
| SQL Server | 0.154 | 0.0 | 0.154 | 0.0 |
| MS-Access | 0.017 | 0.153 | 0.17 | 0.9 |

The final step is to calculate the relative closeness to the ideal solution $C_i^* = S'_i / (S_i^* + S'_i)$ as illustrated in Table 16, above.

With regard to the values of closeness coefficients of the two COTS databases, Microsoft Access $C_i^* = 0.9$, and Oracle 9i $C_i^* = 0.77$, are both compromise solutions. So, Microsoft Access becomes the most dominating alternative having highest $C_i^* = 0.9$ rank, thus Microsoft Access should be selected as the best COTS component among the given four alternatives.

## 9 Comparison of AHP-TOPSIS and AHP-VIKOR

Multi Criteria Analysis (MCA) is appropriate to solve the problem relating to several aspects. AHP-TOPSIS and AHPVIKOR are two ranking methods of MCA. However, these two methods are different in their basic definitions. In this current research, we applied these two methods to find the comprise solution of the alternative COTS database selection and have shown the difference of these methods. The main features of VIKOR and TOPSIS are summarized here in order to clarify the differences between these two methods:

- Procedural basis: Both methods assume that there exists a performance matrix obtained by the evaluation of all the alternatives in terms of each criterion. Normalization is used to eliminate the units of criterion values. An aggregating function is formulated, and it is used as a ranking index. In addition to ranking, the VIKOR method proposes a compromise solution with an advantage rate.
- Normalization:- The difference appears in the normalization used within these two methods. The VIKOR method uses linear normalization and the normalized value does not depend on the evaluation unit of a criterion. The TOPSIS method uses vector normalization and the normalized value can be different for different evaluation units of a particular criterion.
- Aggregation :- The main difference appears in the aggregation approaches. The VIKOR method introduces an aggregating function, representing the distance from the ideal solution. This ranking index is an aggregation of all criteria, the relative importance of the criteria, and a balance between total and individual satisfaction. The TOPSIS method introduces the ranking index, including the distances from the ideal point and from the negative-ideal point. These distances in TOPSIS are simply summed in Tables 14 and 15, without considering their relative importance. However, the reference point could be a major concern in decision making, and to be as close as possible to the ideal is the rationale of human choice. Being far away from a nadir point could be a goal only in a particular situation and the relative importance remains an open question. The TOPSIS method uses n-dimensional Euclidean distance that by itself could represent some balance between total and individual satisfaction, but uses it in a different way than VIKOR, where weight v is introduced.
- Solution:- Both methods provide a ranking list. The highest ranked alternative by VIKOR is the closest to the ideal solution. However, the highest ranked alternative by TOPSIS is the best in terms of the ranking index, which does not mean that it is always the closest to the ideal solution. In addition to ranking, the VIKOR method proposes a compromise solution with an advantage rate.

## 10 Comparison Solutions

The compromise ranking method was applied with data given in Table 4 for AHP-VIKOR and Table 10 for AHP-TOPSIS. The obtained ranking list (by AHP-VIKOR) is presented in Table 9. The ranking results are obtained by applying another method, named AHP-TOPSIS is presented in Table 16.

There are two compromise solutions obtained by AHP-VIKOR, SQL Server 2005, Oracle 9i, are both compromise solutions because the alternative with first position is SQL Server 2005 with Q(SQL Server 2005) = 0.0289, and Oracle 9i is the alternative with second position with Q(Oracle 9i ) = 0.1328 as illustrated in table 9 . As DQ=1/(J-1) = 1/(4-1)= 0.333, so Q(Oracle 9i ) – Q(SQL Server 2005) = 0.1328 – 0.0289 = 0.1039 < 0.333. Which is not satisfied the condition one (C1) in step 5 Q (SQL Server 2005) – Q(Oracle 9i ) >= 0.333, but alternative SQL Server 2005 is the best ranked byQ(Oracle 9i ) –Q(SQL Server 2005) = 0.1328 –0.02890.333, Q(Ingres) –Q(SQL Server 2005) = 0.8230 –0.0289 =0.7941 > 0.333

## 11 Conclusion and Future Works

This study relived that the use of commercial off-the-shelf (COTS) software products in large systems provides many benefits, including rapid delivery to end users. The reuse of software components that are already tested and validated is an opportunity of deploying quality software in the operational environment. For systems that depend on COTS products, the evaluation and selection of appropriate reusable component is essential to the success of the entire system. There are several existing methodologies used to evaluate for adoption COTS components. The ranking results obtained by the AHP-TOPSIS method indicate there are two compromise solutions, Microsoft Access and Oracle 9i because the values of closeness coefficients of Microsoft Access $C_i^*$ = 0.90 and Oracle 9i $C_i^*$ = 0.77 as illustrated in table 16, the most dominating alternative having highest $C_i^*$ = 0.9 is Microsoft Access may be considered as the best compromise solution and the Oracle 9i may be considered as the second-best compromise solution. The result of multicriteria optimization (AHP-VILOR, and AHP-TOPSIS) is that the Oracle 9i is more suitable for database software because it compromises solutions in both methods (AHP-VILOR, and AHP-TOPSIS). As future work, it is open to propose a model that combines both methods to evaluate COTS.

## References


1) Björklund S, Stern O. Managerial differences despite Isomorphic forces: A comparative study of Handelsbanken and Nordea. 2020.
2) Hollands F, Escueta M. How research informs educational technology decision-making in higher education: the role of external research versus internal research. *Educational Technology Research and Development*. 2020;68(1):163–180. Available from: https://dx.doi.org/10.1007/s11423-019-09678-z.
3) Liou JJH, Chuang YC, Zavadskas EK, Tzeng GH. Data-driven hybrid multiple attribute decision-making model for green supplier evaluation and performance improvement. *Journal of Cleaner Production*. 2019;241. Available from: https://dx.doi.org/10.1016/j.jclepro.2019.118321.
4) Ansah RH, Sorooshian S, Mustafa S. Analytic hierarchy process decision making algorithm. *Global journal of pure and applied mathematics*. 2015;11(4).
5) Baykasoğlu A, Gölcük İ. Development of a novel multiple-attribute decision making model via fuzzy cognitive maps and hierarchical fuzzy TOPSIS. *Information Sciences*. 2015;301:75–98. Available from: https://dx.doi.org/10.1016/j.ins.2014.12.048.
6) Opricovic S. Multicriteria optimization of civil engineering systems. Belgrade. 1998.
7) Yurin AY. Group decision-making methods for adapting solutions derived from case-based reasoning. . *Scientific and Technical Information Processing*. 1982;42(5):375–381. Available from: https://doi.org/10.3103/S014768821505010X.
8) Rawashdeh A, Matalkah B. A New Software Quality Model for Evaluating COTS Components. *Journal of Computer Science*. 2006;2(4):373–381. Available from: https://dx.doi.org/10.3844/jcssp.2006.373.381.
9) Yadav S, Kishan B. Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software. *International journal of emerging trends in engineering research*. 2020;8(6).
10) Epstein RH, Dexter F. Database Quality and Access Issues Relevant to Research Using Anesthesia Information Management System Data. *Anesthesia & Analgesia*. 2018;127:105–114. Available from: https://dx.doi.org/10.1213/ane.0000000000003324.
11) Abramatic JF, Cosmo RD, Zacchiroli S. Building the universal archive of source code. *Communications of the ACM*. 2018;61(10):29–31. Available from: https://dx.doi.org/10.1145/3183558.
12) Gridin VN, Dmitrevich GD, Anisimov VI, Vasiliev SA. Methods of constructing service-oriented computer-Aided circuit design systems based on WebSocket Protocol and Oracle database. *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. 2017;p. 409–412.
13) Mukherjee S. Indexes in Microsoft SQL Server. *arXiv preprint* . 2019. Available from: arXiv:1903.08334.
14) Zhang P. Practical Guide to Large Database Migration. CRC Press. 2019.
15) Ma F, Shi W, Yuen KF, Sun Q, Guo Y. Multi-stakeholders' assessment of bike sharing service quality based on DEMATEL-VIKOR method. *International Journal of Logistics Research and Applications*. 2019;22(5):449–472.
16) Hamdy A, Elsayed M. Towards more accurate automatic recommendation of software design patterns. *Journal of Theoretical & Applied Information Technology*. 2018;96(15).