Trade-offs Based Decision System for Adoption of Cloud-Based Services

Radhika Garg(✉), Marc Heimgartner, and Burkhard Stiller

Communication Systems Group CSG@IfI, University of Zürich, UZH,
Binzmühlestrasse 14, 8050 Zürich, Switzerland
{garg,stiller}@ifi.uzh.ch, marc.heimgartner@uzh.ch

Abstract. Decision of adopting any new technology in an organization is a crucial decision as it can have impact at the technical, economical, and organizational level. Therefore, the decision to adopt Cloud-based services has to be based on a methodology that supports a wide array of criteria for evaluating available alternatives. Also, as these criteria or factors can be mutually interdependent and conflicting, a trade-offs-based methodology is needed to take a decisions.

This paper discusses the design, implementation, and evaluation of the prototype developed for automating the extended theoretical methodology of Trade-offs based Methodology for Adoption of Cloud-based Services (TrAdeCIS) developed in [5]. This system is based on Multi-attribute Decision Algorithms (MADA), which selects the best alternative, based on priorities of criteria of a decision maker. The applicability of this methodology to the adoption of cloud-based services in an organization is validated with a use-case and is even extended to other domains, especially for Train Operating Companies.

Keywords: Cloud computing · Cloud adoption · Decision support system · Multi-attribute Decision Algorithms

1 Introduction

Traditional IT (Information Technology) aligns resources according to applications in order to fulfill their business requirements. Each application has its own dedicated infrastructure and data storage [11]. For data protection and continuity of business operations, dedicated backup and recovery solutions are also deployed. As an alternative, Cloud Computing (CC) has recently emerged as a paradigm that offers its users the flexibility of scaling their computing resource usage without the concern of over or under-provisioning. CC is the result of evolution and embracement of various technologies as that of Virtualization (separating physical devises into one or more virtual devices), Service-oriented Architecture (based on loosely coupled independent services), and Utility Computing (which charges the user based on the usage instead of a fixed rate). The major benefits of cloud-based services include pay-as-you-go model, business agility and flexibility, increase in economies of scale. However, there also exist disadvantages

© Springer International Publishing AG 2017
M. Helfert et al. (Eds.): CLOSER 2016, CCIS 740, pp. 65–87, 2017.
DOI: 10.1007/978-3-319-62594-2_4

in terms of security, privacy risk, or vendor-lock in [1]. CC has four deployment models (1) Private Cloud, (2) Public Cloud, (3) Hybrid Cloud, and (4) Community Model [1]. CC today can be delivered as XaaS (Anything-as-Service), which includes the fundamental service models of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) and can be extended to anything such as Network-as-a-Service, Database-as-a-Service, or Communication-as-a-Service, Business-as-a-Service [5]. Owing to several available options an organization has to decide various following aspects:

- **Selection of Deployment Model:** Each deployment model has its advantage and disadvantages; therefore, several factors have to be considered while making a decision.
- **Selection of Service Model:** Each service model consists of various requirements to be fulfilled both from the side of Cloud Service Provider (CSP) and the organization that plans to adopt the solution. For example, in case of PaaS, CSP provides both hardware and software on which applications run, whereas, in IaaS a virtual machine is provided by CSP. For OS and middleware, organization is responsible. Therefore, here again the decision of which service model can be adopted depends on various requirements.
- **Selection of Appropriate Service Package:** Also, there is a variation in terms of capabilities CSP provider in numerous different packages. These packages can have different benefits or drawbacks. For example, some CSPs might offer services at low cost, however, they might then not offer backups or redundant storage of data at multiple locations. This implies that the factors influencing the decision can be dependent and mutually contradictory. Therefore, organization has to make a trade-off and make the selection based on the best match to its requirement.

Due to this wider range of decisions to be taken and selections to be made, an automated Decision Support System with industrial strength will have to make trade-off decisions, which need to show a respective detailed evaluation of alternative options. Thus, the research questions to be answered are the following:

- How can a quantified trade-off based strategy be established?
- How can such a strategy evaluate several alternatives with respect to numerous interdependent and contradictory requirements?

To address this problem of decision making while adopting Cloud-based services in an organization, the methodology TrAdeCIS was introduced [5]. TrAdeCIS automates the decision process and the paper evaluates its applicability and validity not only in the context of Cloud Computing but also in the decision of adopting any new technology in an organization.

The remainder of this paper is structured as follows. Section 2 discusses related work in the field of the decision analysis for adopting any technology in an organization. It also highlights existing gaps and how TrADeCIS bridges them. Section 3 presents the architecture of the prototype for implementing TrAdeCIS and discusses the applicability and relevance of the algorithms used for making

such a decision. Section 4 presents key functionality and tests as well as evaluates the methodology with respect use cases from the domain of cloud computing and Internet on train. Furthermore, this section also evaluates the performance of the implementation of the prototype. Finally, Sect. 5 concludes the paper.

2 Related Work

Spokesperson of Gartner stated that customers should be very careful while selecting the correct service provider, and ask them detailed questions about contractual terms [10]. Therefore, the decision maker has to be aware of complete requirements, their interdependencies, and conflicts in order to evaluate different CSPs. This was performed in [6]. The second challenge is to develop a quantitative approach to make decision of adopting best alternative that encompasses all requirements (criteria) and their interrelations. There have been efforts in the past to make a decision whether to move the legacy infrastructure into cloud or not. [1] and [14] propose two different approaches. While [1] compares the cost of using a cloud-based service with the costs of a datacenter on an hourly basis, [14] presents an approach to compare the costs of leasing and purchasing a CPU (Central Processing Unit) over several years. Both of these approaches only consider cost as a factor, when there are multiple conflicting factors that must be considered. Also, this approach is not open to an extension to multiple quantitative factors (that can have different measurement units) and to factors that are of qualitative nature [9]. Therefore, there is a need for a methodology encompassing multiple factors for evaluating several alternatives.

In the past, Multi-attribute Decision Algorithms (MADA) have been used for the decision on outsourcing [15] that supports multiple factors. MADAs include a finite set of alternatives, and their performance in multiple criteria is identified in the beginning of the analysis. These methods can either be used to sort or classify available alternatives. However, the current research is restricted to a number of predefined factors for taking a decision. Research so far on a cloud adoption decision process also suggests approaches such as Goal-oriented Requirements Engineering (GRE) ([2,16]) and a quantified method using MADA [9,13]. GRE-based approaches are based on a step-by-step process of fulfilling requirements of the cloud user and are qualitative in nature. MADA-based approaches are quantitative in nature, however, fail to evaluate impact such an adoption will have on an organization and do not incorporate business or organizational aspects in the decision. They also do not consider the influence of one attribute over another. In addition, they do not establish a trade-off strategy, where conflicting factors are involved. A trade-off strategy refers to the technique of reducing or forgoing one or more desirable parameters in exchange of increasing or obtaining other desirable outcomes in order to maximize the total return.

As shown in Table 1, a gap still exists in terms of not only developing a trade-offs-based methodology for decision making while adopting CC, but also in automating it. The comparison of related work to the work done in this paper is based on four key features, "$\surd$" describing the presence and "$\times$" denoting

Table 1. Comparison of related work with respect to main characteristics.

Features	Cost-based approaches	MADA-based approaches	TrAdeCIS
Interrelations of factors	Partially	Partially	√
Trade-offs based quantified methodology	×	×	√
Automated decision support system	×	×	√
Applicability to other domains	×	Partially	√

the lack of that feature. This paper, therefore, fills this gap by (a) automating trade-off based quantified methodology and (b) studying its applicability for a CC use-cases, which models all relevant factors and their interrelations.

3 Research Methodology and Architecture of the System

The methodology followed to establish trade-offs-based decision of selecting the best alternative is based on algorithms of MADA- The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and Analytic Network Process (ANP). The methodology of TrAdeCIS is shown in detail with the flow diagram in Fig. 1. Both of these algorithms of ANP and TOPSIS require multiple alternatives and criteria as inputs. TOPSIS is used to rank the alternatives from the technical perspective. ANP is used to rank the same alternatives from economical and business perspective. The relevant criteria from the domain of CC, has already been identified in [6]. A user can either select the relevant criteria from this list, or enter their own requirements. The details of these algorithms and their implementations are described in the following sections. Furthermore, the architecture as implemented for the prototype of TrAdeCIS and the database model of the system developed is also discussed below.

3.1 TOPSIS

TOPSIS is based on the concept that the optimal solution is the one, which has geometrically the shortest distance from the best possible solution and the longest distance from the worst possible solution [8]. TOPSIS does pair-wise comparisons of all alternatives across all the criteria and facilitates trading-off a poor performance of an alternative in one factor by a good performance in another factor. Pair-wise comparison is a process of comparing alternatives in pairs to judge which of the two alternatives is preferred, has better performance with respect to a factor, or whether or not the two alternatives are performing at the same level with respect to a factor. Listing 1, expects three inputs:

– a N × M matrix of the values with N criteria as columns and M alternatives as rows.

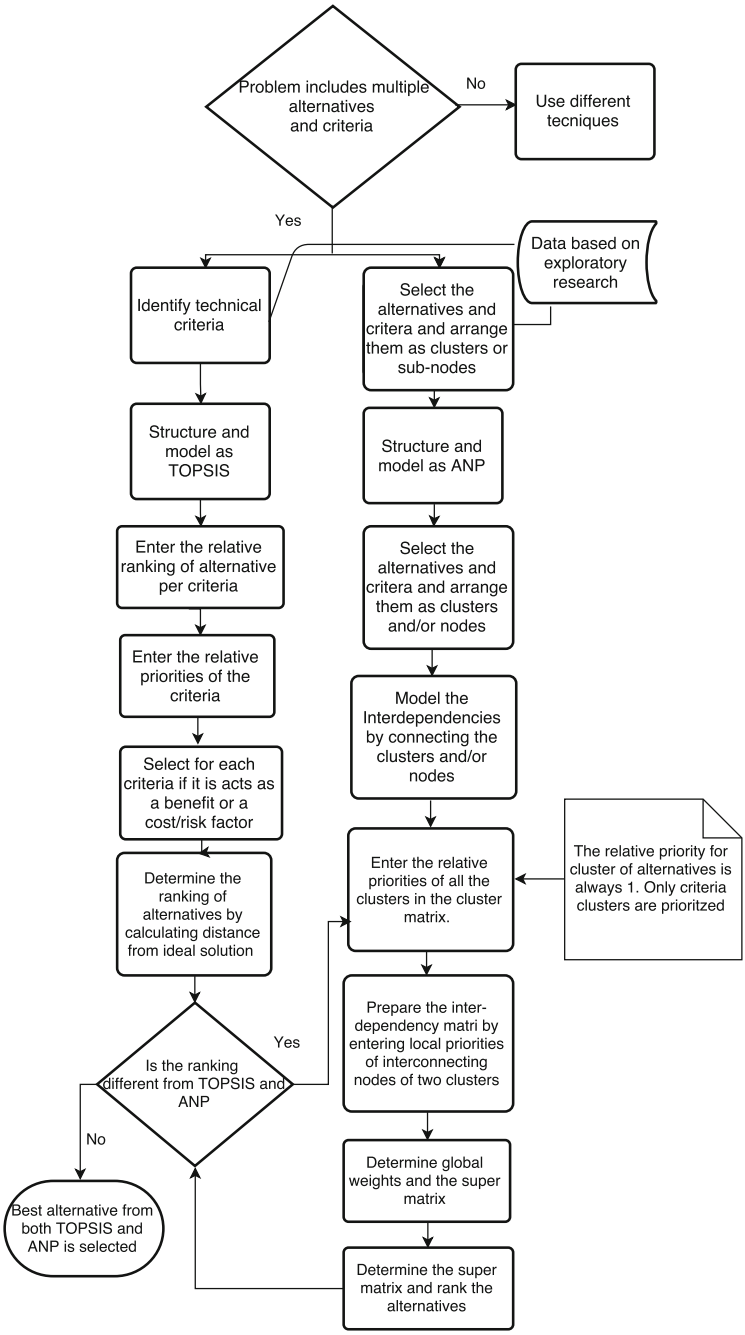

Fig. 1. Flow diagram for TraAdeCIS.

– weights, N priority values in order to prioritize the criteria.
– has_positiv_effect, N true or false values, depending on the positive or negative impact of the criteria on the decision.

As shown in Listing 1, the first step is to normalize the NxM matrix and weights in order to gain homogeneous values, which can be mutually compared (Line 4). This step is performed to transform the attributes having different dimensions into non-dimensional attributes, hence allowing comparisons across criteria. From the normalized and weighted matrix the minimum and maximum values are taken for each criteria for later use of finding the relative distance of alternative from best possible solution and worst possible solution (Line 7, 8). After that the best possible solution is computed by taking the maximum value, if the criteria has a positive effect on the result or the minimum value, if it has a negative impact on the result (Line 11).

Listing 1. Algorithm for TOPSIS as Implemented in TrAdeCIS.

```
def topsis(matrix, weights, has_positiv_effect , normalization=vector_normalization):

    # normalize and apply weights
    weighted_matrix = normalization(weights) * normalization(matrix)[1]

    # extract min and max values for each column
    mins = numpy.min(weighted_matrix, axis=0)[2]
    maxs = numpy.max(weighted_matrix, axis=0)[2]

    # create ideal and anti ideal arrays
    ideal = numpy.where(has_positiv_effect, maxs, mins)[3]
    anti_ideal = numpy.where(has_positiv_effect, mins, maxs)[3]

    # calculate distances to the ideal and anti ideal arrays
    distance_ideal = norm(weighted_matrix - ideal, axis=1)[1]
    distance_anti_ideal = norm(weighted_matrix - anti_ideal, axis=1)[1]

    # compute relative closeness
    relative_closeness = distance_anti_ideal / (distance_ideal + distance_anti_ideal)[4]

    return relative_closeness
```

[1]Number of executions: $N * (M + (M-1) + M^{1/2})$, for N columns the M values are squared, summed and then the square root of the sum is taken (vector normalization)
[2]Number of executions: $M * N$, for N columns check the values
[3]Number of executions: N, create arrays with N values
[4]Number of executions: $2 * N$, N additions and divisions

The worst possible solution is constructed by taking the minimum value if the impact is positive and the maximum value if the impact is negative (Line 12). The next step is to compute the distance of the matrix to the ideal as well as the anti-ideal solution. This is done by computing the Euclidean distance (Line 15, 16). Finally the relative closeness is computed. Ranking of alternatives is based on the relative closeness of alternatives to the ideal solution. Higher the value, higher is the ranking of the alternative (Line 19). The complexity[1] of the TOPSIS algorithm,

[1] Complexity: $N * (M + (M-1) + M^{1/2}) + 2 * M * N + 2 * N + 2 * (N * (M + (M-1) + M^{1/2}) + 2 * N = 8 * M * N + N + 3 * M^{1/2} = O(M * N)$.

with respect to implementation shown above is $O(N * M)$ where N is the number of alternatives and M the number of criteria.

3.2 ANP

ANP is the generalization of the Analytic Hierarchy Process (AHP) [8], and is a method where dependencies can be modeled between any of the elements. In ANP, all criteria are represented as a cluster, and their sub-criteria (if any) are modeled as elements or nodes within that cluster. Also, all available alternatives constitute an additional cluster. Each connection symbolizes the interdependency between the 2 connected nodes or clusters. On one hand, it results in the modeling of more accurate models, but on other hand it increases the complexity of the required input. This is because pairwise comparison matrices where criteria or alternatives are compared with respect to every other interconnected element in the network(see use cases for an example) has to be constructed. As shown in Listing 2 the next step is to compute the super matrix. Super matrix is generated with the eigenvectors of all the possible pairwise comparison matrices.

Listing 2. Generation of the Super Matrix as Implemented in TrAdeCIS.

```
supermatrix: function (clusterNodes) {
    var children = graph.findChildren(clusterNodes);
    var matrix = utils.matrix(children.length, 0);

    children.each(function (column, sourceNode) {
      children.each(function (row, targetNode) {
        matrix[row][column] = graph.getValue(sourceNode, targetNode) || 0;
      });
    });

    return matrix
  }
```

As the super matrix represents all interrelations between any two nodes in the network, the alternatives are listed always as last n elements of the super matrix, where n is the number of alternatives. The ranking of alternatives in obtained by computing the limit matrix, and transforming it into an array. In order to compute the limit matrix, the super matrix has to be raised to high odd powers until it converges. It can be shown that the limit exists if the matrix is column stochastic [12].

As shown in Listing 3, the computation of limit matrix is an iterative process where the matrix is raised by the power of 3 and then again normalized in order to keep the matrix column stochastic (Line 7). Then the result is checked if it is equal up to the 8th decimal precision with the previous result (Line 12). If so the process is ended. Usually this takes around 3 iterations to find a result. The number of iterations depends on the limit of the super matrix (the power at which the matrix converges), and therefore on the values in the super matrix, which consist of the global cluster comparison, the criteria comparison of the cluster, and the criteria value.

Listing 3. Generation of Limit Matrix as Implemented in TrAdeCIS.

```
def limit_matrix(matrix):
    result = matrix
    previous_matrix = result

    while True:
        result = linear_normalization(numpy.linalg.matrix_power(result, 3))[1]

        if numpy.isnan(numpy.sum(result))[2]:
            raise ArithmeticError('received not a number')

        if numpy.allclose(previous_matrix, result)[3]:
            break

        previous_matrix = result

    return result
```

[1]Number of executions: $2 * N^3$, two matrix multiplications are done, matrix multiplication has a complexity of $O(N^3)$

[2]Number of executions: $M * N$, all the values are summed up

[3] Number of executions: $M * N$, all the values are compared

The reason for raising the super matrix by the power of 3 is that odd numbers have the advantage of preserving the structure of the matrix (in matrix multiplication, depending on where a zero is the other values might switch places with the zeros). When the limit is found, the values for the whole row are the same. The advantage, however, is that if is raised by an odd number the first column will certainly have non-zero values. These values denote the ranking of the alternatives. The value of 3 is chosen so as to maintain a balance between the rising complexity of the computation with higher values, and the number of iterations needed to compute the limit matrix. The complexity[2] of the algorithm is $O(N^3)$, where N is the dimension of the super matrix.

3.3 Trade-off Based Decision

A trade-offs-based strategy is required, if the ranking of alternatives obtained by using TOPSIS (from technical perspective) and ANP (from business – economical and organizational – perspective) is different. TrAdeCIS therefore, as shown in Fig. 1, compares the rankings obtained and gives the option to a decision maker to select the best technical solution at a trade-off of business value. Trade-offs are achieved by altering the priorities of criteria. There are essentially three possible dimensions at which priorities can be adjusted in ANP.

1. At the global cluster level, prioritizing an entire cluster compared to others. The alternatives cluster can also be compared as an exception to other clusters if needed.

[2] Complexity: $2 * N^3 + M * N + M * N = O(N^3)$.

2. At the cluster level, comparing the importance of criteria in a cluster.
3. At the criteria level, changing the values of the comparison matrix.

In order to assist a decision maker in deciding which criteria or alternative should be adjusted, TrAdeCIS provides a basic approach to calculate the impact of changing any of these values. A column of the super matrix shows the influence that a node has on other criteria and alternatives (outgoing influence). Therefore, by increasing and decreasing the row values in a super matrix the influence of a node to the final rankings as per ANP can be evaluated. Hence, the approach developed and followed here for calculating trade-offs is outlined in Algorithm 1. The increase (*inc*) and decrease (*dec*) for each element of original super matrix as shown in Algorithm 1 is calculated using on the current normalized value and half of the minimum element of the original super matrix (*relative_change*). Choosing *relative_change* to be less than the minimum element of the super matrix removes the possibility of division by zero error in *inc*. Also, *dec* will never be zero or negative, because the *relative_change* is smaller than the lowest value in the super matrix.

Table 2. Original super and limit matrices for trade-offs.

(a) Super matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.75	0.3333	0	0
A2	0.25	0.6667	0	0
(b) Limit matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.5417	0.5417	0	0
A2	0.4583	0.4583	0	0

The above developed methodology is illustrated here with an example to understand the application of this trade-offs process. In this example the result of TOPSIS favors A2, while ANP favors A1. A user now has to make a trade-off decision by adjusting ANP values so that the result will also be A2. By analyzing the super matrix in Table 2(a) and comparing it with the model structure shown in Fig. 2 the meaning of values in the super matrix can be analyzed.

Each row of the super matrix represents how much a node is influenced by other criteria or alternatives (incoming influence), *e.g.,* the first row of Table 2(a)

Algorithm 1. Algorithm for establishing Trade-offs.

▷ % S_x denotes the super matrix x
▷ %: L_x denotes the limit matrix of super matrix x
▷ %: *Row* is the set of elements in a row of a super matrix or limit matrix
▷ %: *Node* denotes to a criteria or alternative
▷ %: *inc* denotes the value by which values of super matrix has to be increased
▷ %: *dec* denotes the value by which values of super matrix has to be decreased
▷ %: *current_value* denotes an element of original super matrix
▷ %: *relative_change* = 0.5 ∗ (minimum element of the original super matrix)

Compute the limit matrix L_c from the current super matrix S_c

for each row in the super matrix **do**
  $inc = relative\_change/(1 - (current\_value + relative\_change))$
  $dec = relative\_change/(1 + (current\_value - relative\_change))$
  Construct super matrix S_{row^+} by increasing each element in *Row* by *inc*
  Construct super matrix S_{row^-} by decreasing each element in *Row* by *dec*
end for

for each constructed super matrix ∈ $\{S_{row^+}, S_{row^-}\}$ **do**
  Compute the limit matrix L_{row^+}
  Compute the limit matrix L_{row^-}
end for

for each computed limit matrix ∈ $\{L_{row^+}, L_{row^-}\}$ **do**
  Compute the differences to the original limit matrix L_c, $L_{row^+} - L_c$ and $L_{row^-} - L_c$
end for
Sort the the differences of the limit matrix in the previous step to obtain a list of the nodes which have the highest impact

for each positive difference of the limit matrix **do**
  Repeat the process but only specific to one element at a time
end for

shows that C1 is equally influenced by A1 and A2 and not influenced by C1 and C2 at all. In Table 2(a) the minimum value is 0.25, therefore the *relative_change* is 0.125. The algorithm is applied here only specific to row 3, for illustration purposes.

Tables 3(a) and 4(a) show the super matrices after increasing and decreasing the non-zero values of row 3 by *inc* and *dec* respectively. By computing the limit

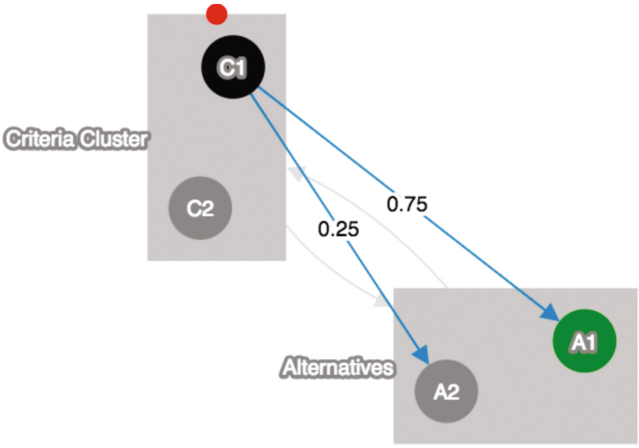

Fig. 2. ANP model for trade-off.

Table 3. Super and limit matrices with increase for trade-offs.

(a) Super matrix				
	C1	C2	A1	A2
C1	0	0	0.35	0.35
C2	0	0	0.25	0.25
A1	1.75	0.5641	0	0
A2	0.25	0.6666	0	0
(b) Limit matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.7163	0.7163	0	0
A2	0.2837	0.2837	0	0

matrix for these super matrices (cf. Tables 3(b) and 4(b)), and subsequently calculating the difference to the original limit matrix (cf. Tables 5(a) and (b)) the influence of A2 on the result of ANP can be identified.

This process is repeated for each row and the values corresponding to A2 (as this was the highest ranked alternative according to TOPSIS) from limit matrices are shown in Table 6(a). Table 6(a) shows that increasing A2 in terms of its interconnected node is the most beneficial (highest value corresponds to A2) for ranking A2 as the highest ranked alternative per ANP. Table 6(b) shows for every positive value in Table 6(a) those values obtained by increasing or decreasing only one element at time in the original super matrix. For example,

Table 4. Super and limit matrices with decrease for trade-offs.

(a) Super matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.4167	0.1754	0	0
A2	0.25	0.6666	0	0
(b) Limit matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.3924	0.3924	0	0
A2	0.6076	0.6076	0	0

Table 5. Differences of limit matrices.

(a) Difference of increased matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	0.1746	0.1746	0	0
A2	−0.1746	−0.1746	0	0
(b) Difference of decreased limit matrix				
	C1	C2	A1	A2
C1	0	0	0.5	0.5
C2	0	0	0.5	0.5
A1	−0.1492	−0.1492	0	0
A2	0.1492	0.1492	0	0

as C2 has positive value in Table 6(a), two values are obtained in Table 6(b) corresponding to the interrelation of C2 to A1 and C2 to A2. This leads to the identification of the node that is interrelated to A2: when changed in terms of its associated priority it will make A2 the highest ranked alternative. In this example, as A2 associated with C2 has highest positive value, change in priority of C2 will lead to the desired ranking.

Table 6. Trade-offs result.

(a) Row specific limit matrix values		
Node	Increase	Decrease
C1	−0.0520	0.0520
C2	0.0520	−0.0520
A1	−0.1746	0.1492
A2	0.1548	−0.1421

(b) Element specific limit matrix values			
Node	Connected node	Increase	Decrease
C1	A1	-	0.0246
C1	A2	-	0.0226
C2	A1	0.0297	-
C2	A2	0.0285	-
A1	C1	-	0.0917
A1	C2	-	0.0392
A2	C1	0.0492	-
A2	C2	0.1250	-

3.4 Implementation Architecture

TrAdeCIS 1.0 was implemented as a traditional client-server architecture, where for each action a new request is made and the server answers with the corresponding markup. This architectural implementation lead to duplication of code, because some parts of the webpage had to update asynchronous. The further details of TrAdeCIS 1.0 are available in [4,7]. For TrAdeCIS 2.0 the implemented architecture was designed based on the Single-Page-Application (SPA) architecture. A SPA is defined as web application or web site that fits on a single web page. This can be either achieved by initially loading all necessary code for the representation (HTML, JavaScript and CSS) or resources are dynamically loaded and added to the page as necessary. This design has been implemented in order to improve performance as well as avoid potential duplication of code. A cycle of requests in a tradition client-server architecture with asynchronous elements (*e.g., TrAdeCIS 1.0*) can be outlined like the following:

1. Client visits the webpage
2. Backend processes the request, renders the webpage via a template language
3. Client receives the webpage
4. Client changes data on the previously received webpage
5. Frontend issues an AJAX request and updates the user interface

This approach shows overlapping logic at step 2 and 5 just in different environments (frontend, backend) and programming languages. A SPA architecture

has the advantage to overcome those issues by separating all logic for the representation on the frontend and all the data manipulation logic on the backend. The added advantages are (1) the clear separation of concerns for the frontend and backend, (2) that user no longer experiences browser reloads (everything will now be loaded asynchronously by the client) and that (3) the data flow is minimized (only the necessary data is loaded, no markup). Additionally future implementations of native clients would be simplified as they have already an API to address. However this approach has also disadvantages, indexing will no longer work, *e.g.,* from Google and also because routing has to be done in the frontend, therefore a request on a route in a web browser will not exist. There is one way to solve both problems by using the concept of "universal javascript". Briefly said it means that javascript code will also run in the backend. In order that this is possible a node.js instance has to process the JavaScript code and then return a html page. However this will raise the complexity of the backend because the communication between the two instances has to be managed. Due to this added complexity TrAdeCIS does currently not implement a "universal javascript" approach. In TrAdeCIS 2.0 the implemented architecture reads as follows:

- The backend is built with Django and Django REST framework and exposes REST endpoints for all the necessary tasks.
- The frontend is built with React.js, consumes the data from the REST endpoints and visualizes them on the client side.

Backend. The architecture of the system follows the community standards with Django projects. Django is fullstack web framework for Python. In Django coherent logic is bundled in a so-called "app". TrAdeCIS is built with two

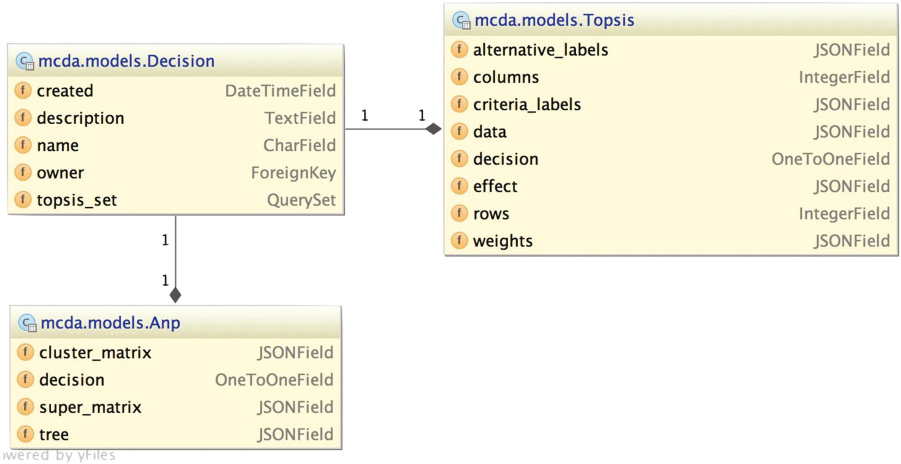

Fig. 3. Database interrelations.

apps: (1) "mcda", for storing, computing and visualizing TOPSIS and ANP, and (2) "account", to manage the different access levels which TrAdeCIS provides. The app "mcda" consists of three database models namely, Decision, TOPSIS and ANP (cf. Fig. 3). Decision model denotes one use-case, which consists of a name, optional description and the data for TOPSIS and ANP, which are stored in their respective tables. The access as well as the modification of the data is exposed via REST endpoints by utilizing the "Django REST Framework", which simplifies the creation of REST APIs with Django.

Frontend. The SPA frontend is built with several new technologies, which are briefly introduced: (1) "React.js", is a Frontend-Framework developed and maintained by Facebook. It challenges current standards by writing everything in Javascript. Rather than DOM changes it utilizes a virtual DOM which is performs faster on changes and computes the minimal changes which have to be done to the actual DOM. The key feature is that a webpage is built with reusable components, which are defined by programmer. There are many other Frontend-Frameworks which could be used and there is no specific reason to favor React.js. (2) "Webpack", is a module bundler, which allows to build browser javascript which can be modularized. Apart from that there are many additional features which make Webpack valuable, for example minification of code, import of other files (e.g. css), removing unused code, etc. (3) "Redux", is a state management library which makes the state immutable at all time and therefore leads to clearer, as well as better testable state. Additional new technologies and concepts like "Css Modules", "PostCss", "ES6", "JSX" were used as well as the following libraries "React-Router", "Cytoscape" and "Chartjs".

4 Testing and Evaluation

This section tests the developed system with the objective of evaluating its applicability, usability in various use-cases of making such decisions. The performance values of all alternatives with respect to criteria is taken from [3], which is platform to measure and monitor these values. In addition, performance of the system is also evaluated, which includes calculation of load-up and processing time of the system, depending of number of alternatives and criteria.

4.1 Decision of Adopting PaaS (Use Case 1)

For Use Case 1 (UC1) the scenario of adopting PaaS is considered, with alternative providers and criteria as shown in Table 7. For TOPSIS the criteria of "Runtimes" denotes the number of supported programming languages. "Services" are additional services that are supported (for example databases), and "Add-ons" are additional other programs which can be used. Also, "uptime" of the service in the past 30 days for all the providers is included. In this scenario as well, all these criteria have a positive impact and are weighted equally. The result of TOPSIS is computed with code snippet 1 and is "Heroku".

Table 7. Decision of adopting PaaS input for TOPSIS.

Alternatives	Uptime	RAM (MB)	Runtimes	Services	Add-ons
Heroku	99.91	512	9	2	17
dotcloud	99.95	32	5	1	7
AppHarbor	99.99	512	1	3	33

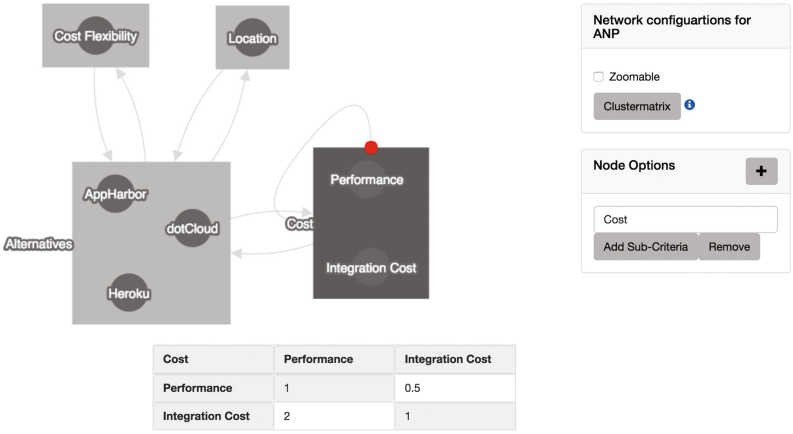

Fig. 4. Decision of adopting PaaS ANP model.

Table 8. Decision of adopting PaaS resulting super matrix.

	Location	Performance cost	Integration cost	Cost flexibility	Heroku	dotcloud	AppHarbor
Location	0	0	0	0	0.083	0.083	0.083
Performance cost	0	0	0	0	0.021	0.056	0.028
Integration cost	0	0	0	0	0.062	0.028	0.056
Cost flexibility	0	0	0	0	0.083	0.083	0.083
Heroku	0.100	0.037	0.104	0.035	0	0	0
dotcloud	0.050	0.025	0.022	0.144	0	0	0
AppHarbor	0.100	0.022	0.040	0.071	0	0	0

For ranking the alternatives from the business perspective the model in Fig. 4 for ANP is constructed. In this case there is a self-loop on the cost cluster, which allows to give relative priority to each criteria in a cluster. Here the criteria of Integration Cost is considered 2 times more important than that of Performance Cost. For this scenario, the resulting super matrix is shown in Table 8 and not every pairwise comparison matrix.

Again by applying Listing 3 the limit matrix is found, shown in Table 9, which ranks dotCloud the highest.

Table 9. Decision of adopting PaaS resulting limit matrix.

	Location	Performance cost	Integration cost	Cost flexibility	Heroku	dotcloud	AppHarbor
Location	0	0	0	0	0.333	0.333	0.333
Performance cost	0	0	0	0	0.140	0.140	0.140
Integration cost	0	0	0	0	0.193	0.193	0.193
Cost flexibility	0	0	0	0	0.333	0.333	0.333
Heroku	0.335	0.335	0.335	0.335	0	0	0
dotcloud	0.343	0.343	0.343	0.343	0	0	0
AppHarbor	0.322	0.322	0.322	0.322	0	0	0

Table 10. Decision of adopting PaaS tradeoff.

Node	Connected node	Increase	Decrease
Location	Heroku	0.00087	-
Location	dotcloud	0.00087	-
Location	AppHarbor	0.00079	-
Performance cost	Heroku	0.00131	-
Performance cost	dotcloud	0.00146	-
Performance cost	AppHarbor	0.00118	-
Integration cost	Heroku	0.00560	-
Integration cost	dotcloud	0.00418	-
Integration cost	AppHarbor	0.00458	-
Cost flexibility	Heroku	-	0.00514
Cost flexibility	dotcloud	-	0.00437
Cost flexibility	AppHarbor	-	0.00427
Heroku	Location	0.01561	-
Heroku	Performance cost	0.00704	-
Heroku	Integration cost	0.01535	-
Heroku	Cost flexibility	0.01089	-
dotcloud	Location	-	0.00643
dotcloud	Performance cost	-	0.00294
dotcloud	Integration cost	-	0.00353
dotcloud	Cost flexibility	-	0.01148
AppHarbor	Location	-	0.00799
AppHarbor	Performance cost	-	0.00264
AppHarbor	Integration cost	-	0.00376
AppHarbor	Cost flexibility	-	0.00656

However, now the results of TOPSIS and ANP do not match and therefore a trade-off is necessary to match the results of TOPSIS and ANP. Table 10 shows the trade-offs result based on the approach shown in Sect. 3.3. These values conclude that the highest change towards "Heroku" can be achieved by increasing the priority of the criteria Location (highest positive value is 0.01561 in Table 10).

4.2 Applicability of TrAdeCIS in Other Domains (Use Case 2)

TrAdeCIS was developed to support organizations in the adoption of cloud-based services. However, organizations may also utilize TrADeCIS to improve their understanding of the value of technologies from other domains than cloud-based services. This is illustrated by applying TrAdeCIS to Train Operating Companies (TOC), who need to take a decision of choosing the best technology, to improve both voice- and data coverage on-board trains (UC2). This decision takes the perspective of the TOC, who is hoping to sell more tickets by providing the service. For the train-to-wayside connection, all on-board solutions are assumed to use the same technology, especially a connection to mobile base stations by 3G or beyond. The following alternatives are considered (cf. Table 11) to be installed on-board of trains:

- Option 1: Wireless Access Point (WAP)
- Option 2: Analog repeater
- Option 3: Femtocells

Technical requirements from these and their relative priorities read:

- Internet has to be available to all passengers with a mobile device (Priority 1)
- Quality of voice calls needs to be improved for all passengers with a phone (Priority 2)
- Internet speed should be as high as possible (Priority 3)

Therefore, after applying TOPSIS to these technical requirements, installation of WAPs is ranked the highest. From the financial/economic requirements perspective, ANP is used to model it as shown in Fig. 5. The factor of Net Present Value, which should be positive as soon as possible, is of highest priority. However, it is broken into sub-factors as that of low deployment time, high revenue,

Table 11. Applicability of TrAdeCIS in other domains input for TOPSIS.

Alternatives	Internet availability	Voice coverage	Internet speed
Option 1	3	1	3
Option	2	2	2
Option 3	2	2	2

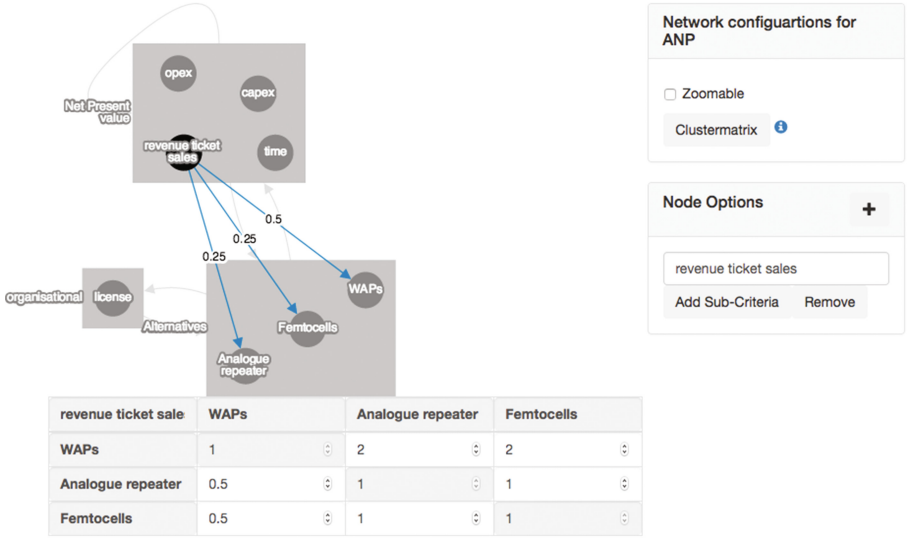

Fig. 5. Applicability of TrAdeCIS in other domains ANP model.

Table 12. Applicability of TrAdeCIS in other domains resulting super matrix.

	OPEX	CAPEX	Time	Revenue ticket sales	License	WAPs	Analogue repeater	Femtocells
OPEX	0	0	0	0	0	0.083	0.083	0.083
CAPEX	0	0	0	0	0	0.083	0.083	0.083
Time	0	0	0	0	0	0.083	0.083	0.083
Revenue ticket sales	0	0	0	0	0	0.083	0.083	0.083
License	0	0	0	0	0	0.333	0.333	0.333
WAPs	0.05	0.041	0.021	0.308	0.425	0	0	0
Analogue repeater	0.05	0.081	0.041	0.154	0.425	0	0	0
Femtocells	0.05	0.027	0.014	0.154	0.142	0	0	0

low capital expenditure, and low operational expenditure. In addition, all these factors contribute differently to the factor of Net Present Value. This is represented with a self-loop and the respect weighing or priorities of these factors are entered in the corresponding comparison matrix. Also in terms of organizational requirements, the TOC prefers to avoid the use of licensed spectrum (medium importance). The resulting super matrix, which is constructed from all comparison matrices, is shown in Table 12. The highest ranked alternative from ANP is also WAPs, as calculated in the limit matrix (cf. Table 13). Therefore, as the ranking obtained from both TOPSIS and ANP is the same, for the scenario of providing internet and voice call connectivity on-board train, WAP is the best alternative.

Table 13. Applicability of TrAdeCIS in other domains resulting limit matrix.

	OPEX	CAPEX	Time	Revenue ticket sales	License	WAPs	Analogue repeater	Femtocells
OPEX	0	0	0	0	0	0.125	0.125	0.125
CAPEX	0	0	0	0	0	0.125	0.125	0.125
Time	0	0	0	0	0	0.125	0.125	0.125
Revenue ticket sales	0	0	0	0	0	0.125	0.125	0.125
License	0	0	0	0	0	0.5	0.5	0.5
WAPs	0.428	0.428	0.428	0.428	0.428	0	0	0
Analogue repeater	0.409	0.409	0.409	0.409	0.409	0	0	0
Femtocells	0.164	0.164	0.164	0.164	0.164	0	0	0

4.3 Performance Test

This section analyses the performance of TrAdeCIS 2.0 with respect to how long functionalities of ranking the alternatives and establishing trade-offs need to execute. Additionally, the improvements of the system compared to TrAdeCIS 1.0, where the implemented architecture was based on a traditional client-server architecture are discussed. All the performance tests are executed on a system with a 2.6 GHz Intel Core i5 CPU, 8 GB 1600 MHz DDR3 RAM and a Intel HD Graphics 4000 1536 MB graphics card. The implemented architecture introduced in Sect. 3.4 has a major influence on the overall performance as well as the user experience. This is due to the fact that all needed files for the representation are loaded on the first connection, and only the necessary data has to be retrieved in subsequent requests. This minimizes data flow and results in load-up time for pages in around 100–300 ms. In TrAdeCIS 1.0 depending on the complexity of the ANP model a user had to wait up to 3 s until the page was fully loaded. It can be concluded that based on these implementation specific changes the execution time has improved by a factor of 5–10.

Another crucial performance impact is on the visualization of TOPSIS and ANP model on GUI. While TOPSIS scales well even with growing number of alternatives and criteria, ANP does not. TOPSIS only needs tabular data as input. ANP, however, has a complex model of interrelations which is created by a html canvas, as shown Fig. 5. The highly complex model of ANP, and its corresponding input value limits the size at which it is user friendly to work with. By trial and error it could be evaluated that the number of nodes in the visualization should not be higher than 100, because otherwise the user has to wait longer intervals to be able to interact with the model. Because of changes in the interaction with the canvas the performance has been increased by a factor of 3 compared to TrAdeCIS 1.0. Figure 6 shows average execution time for 1000 executions of ranking alternatives with TOPSIS and ANP with respect to different number of criteria and alternatives. In case of ANP the values are obtained by an average over 1000 computation of random super matrices. The random super matrices are generated by random values in the interval of 0 and 1 with the given dimension from the number of criteria and alternatives. This can

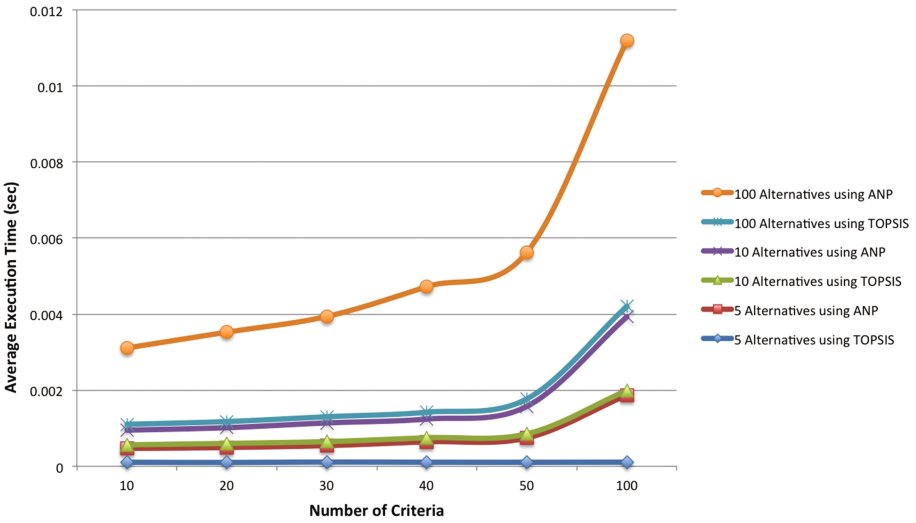

Fig. 6. Execution time measurements of TOPSIS and ANP.

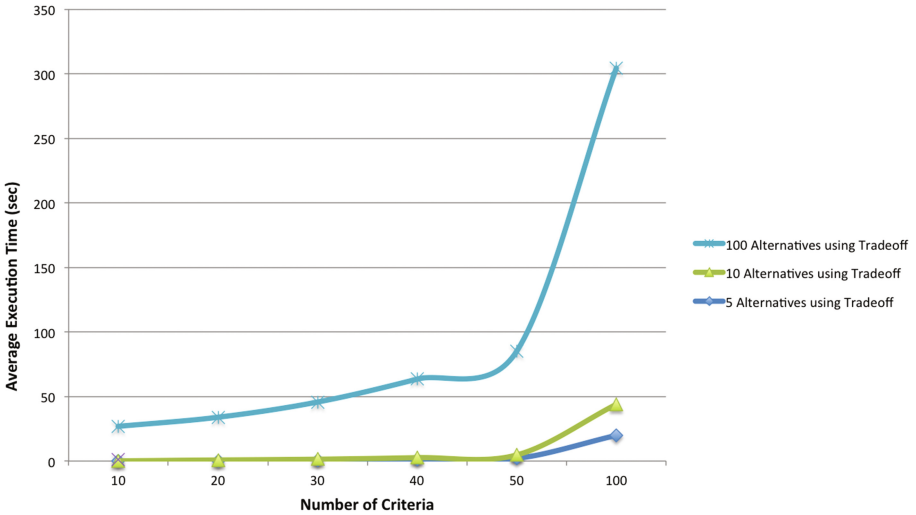

Fig. 7. Execution time measurements of the tradeoff approach.

be justified by the fact that the values are normalized, therefore a higher interval is not necessary. Also when considering the distribution of the random values, zero will occur sparsely, this implies that the resulting random super matrix will be very well connected and therefore computationally more expensive. In general, the execution time should be slightly below the given values since such well connected models are not frequently constructed.

Figure 7 shows the average runtimes for 1000 executions of the trade-off approach introduced in Sect. 3.3. The complexity of the trade-off in a worst case scenario is $O(N^2 * N^3) = O(N^5)$, because for all elements in the NxN matrix the ANP computations have to be executed. However as shown in Sect. 3.3 those are only done for changes with a positive impact on the alternative. In most cases the partition will be around half has a positive and other half an negative impact. Additionally a case where all would have a positive impact is impossible. In the vast majority of cases the complexity will therefore be $O((N/2)^5)$. The results in this section show that the overall performance is in an optimal interval even when complexity of the functionalities of TrAdeCIS is high.

5 Summary and Conclusions

This work has designed, developed, and evaluated the decision support system to automate the methodology of TrADeCIS to facilitate the decision of adopting cloud-based services in an organization. TrAdeCIS makes a trade-offs based quantified decision of selecting the best alternative as per requirements of the organization using integrated MADAs of TOPSIS and ANP. An appropriate use-case (involving train operating companies) validated the applicability of TrADeCIS in a decision process of adopting a different technology besides that of cloud-based services.

The evaluation of the prototype TrAdeCIS 2.0 implemented concluded that TrAdeCIS is scalable to large number of alternatives, factors, and their associated interrelations. This allows modeling of real-world complexities involved in such a decision making. Trade-offs established are measured by change in priorities required in economical and organizational factors, in order to reach the same alternative both from the technical and business perspective. The time taken to establish trade-offs for 50 alternatives and 100 criteria is less than a minute. This is achieved with the optimized implementation of TrAdeCIS to ensure that results obtained are not outdated due to dynamically changing input in performance values of an alternative.

Acknowledgements. This work was partly funded by FLAMINGO, the Network of Excellence Project ICT-318488, supported by the European Commission under its Seventh Framework Program. The authors would also like to thank Bram Naudts for discussions and excellent input with respect to applying TrAdeCIS to Train Operating Companies.

References

1. Armburst, M., Fox, A., Griffith, R., Anthony, J.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)

2. Beserra, P.V., Camara, A., Ximenes, R., Albuquerque, A.B., Mendonça, N.C.: Cloudstep: a step-by-step decision process to support legacy application migration to the cloud. In: IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), Trento, Italy, pp. 7–16 (2012)
3. Cloud Harmony Inc.: Cloud Harmony (2015). https://cloudharmony.com/. Accessed October 2015
4. Garg, R., Heimgartner, M., Stiller, B.: Decision support system for adoption of cloud-based services. In: 6th International Conference on Cloud Computing and Services Science, Rome, Italy, pp. 71–82, April 2016
5. Garg, R., Stiller, B.: Trade-off-based adoption methodology for cloud-based infrastructures and services. In: Sperotto, A., Doyen, G., Latré, S., Charalambides, M., Stiller, B. (eds.) AIMS 2014. LNCS, vol. 8508, pp. 1–14. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43862-6_1
6. Garg, R., Stiller, B.: Factors affecting cloud adoption and their interrelations. In: Proceedings of the 5th International Conference on Cloud Computing and Services Science (CLOSER), SCITEPRESS (Science and Technology Publications, Lda), Lisbon, Portugal, pp. 87–94, May 2015
7. Heimgartner, M.: Design and implementation of prototype for TrAdeCIS. Department of Informatics, University of Zurich, Zurich, Switzerland. Communication Systems Group, December 2015
8. Ishizaka, A., Nemery, P.: Multi-criteria Decision Analysis: Methods and Software. Wiley, Chichester (2013)
9. Menzel, M., Schönherr, M., Tai, S.: $(MC^2)^2$: criteria, requirements and a software prototype for cloud infrastructure decisions. Softw. Pract. Experience **43**(11), 1283–1297 (2013)
10. Moore, S.: Gartner says worldwide cloud Infrastructure-as-a-Service spending to grow 32.8% in 2015e. http://www.gartner.com/newsroom/id/3055225. Accessed October 2015
11. NetApp: The journey from traditional IT to the Cloud-Net App (2015). http://webobjects.cdw.com/webobjects/media/pdf/netapp/NetApp-Virtualization-To-Cloud-Brochure-1.pdf?cm_sp=NAPShowcase-_-Cat4-_-CloudComputing. Accessed October 2015
12. Saaty, T.L., Vargas, L.G.: Decision Making with the Analytic Network Process. Springer, New York (2006)
13. Saripalli, P., Pingali, G.: MADMAC: Multiple Attribute Decision Methodology for Adoption of Clouds. In: IEEE 4th International Conference on Cloud Computing (CLOUD), Washington D.C., USA, pp. 316–323 (2011)
14. Walker, E.: The real cost of a CPU hour. IEEE Comput. **42**(4), 35–41 (2009)
15. Wang, J.J., Yang, D.L.: Using a hybrid multi-criteria decision aid method for information systems outsourcing. Comput. Oper. Res. **34**(12), 3691–3700 (2007)
16. Zardari, S., Bahsoon, R.: Cloud adoption: a goal-oriented requirements engineering approach. In: ACM 2nd International Workshop on Software Engineering for Cloud Computing, Honolulu, Hawaii, USA, pp. 29–35 (2011)