applied sciences

MDPI

Article

QoS-Based Pattern Recognition Approach for Web Service Discovery: Ar_WSDS

Marco Adarme [1,*,†] and Miguel Jimeno [2,†]

[1] Department of Systems and Informatics, Universidad Francisco de Paula Santander, Cúcuta 540003, Colombia
[2] Systems Engineering Department, Universidad del Norte, Barranquilla 080001, Colombia; majimeno@uninorte.edu.co
* Correspondence: madarme@ufps.edu.co or madarme@gmail.com
† These authors contributed equally to this work.

Abstract: Web service composition requires high levels of integration and reliability of the services involved in its operation, which must meet specific quality criteria to ensure their proper execution and deployment. The discovery and selection of web services currently face optimization problems. Many services might satisfy a requirement with similar quality criteria. Because of this, software developers have to choose the most appropriate services for a given composition, complicated by the rapid increase in providers and services available in the cloud. Service composition also implies coupling according to a composition flow and non-functional requirement criteria. Such requirements make selection and composition a complex task not previously solved in the literature. This paper presents Ar_WSDS, a computational approach for web services discovery and selection in cloud environments, which bases its implementation on the brain's pattern recognition systematic functioning. This process allows classifying web services through recognition modules created dynamically based on their quality parameters, resulting in a set of web services suitable for a web service composition. This approach allows a solution to the selection problem using less complex tasks. This paper introduces an architectural and procedural definition that provides the web service description with a pattern to recognize and select services using different recognition levels. We simulated our approach and evaluated it using a dataset from the QWS project that offers a set of quality criteria collected from different providers. The web services are recognized and classified using different quality criteria for the composition and each of their services. The results demonstrate the effectiveness of the discovery and selection process compared to other approaches. Furthermore, Ar_WSDS allows us to recognize and filter out web services with ambiguity and similarity in their provider information, a process that minimizes the discovery space for services.

Keywords: web service composition; cloud computing; pattern recognition

Citation: Adarme, M; Jimeno, M. Qos-Based Pattern Recognition Approach for Web Service Discovery: Ar_WSDS. *Appl. Sci.* **2021**, *11*, 8092. https://doi.org/10.3390/app11178092

Academic Editor: Eui-Nam Huh

Received: 9 July 2021
Accepted: 27 August 2021
Published: 31 August 2021

1. Introduction

Web service composition (WSC) is defined as a logical combination of web services (WSs), which provides a level of collaboration and data exchange between them in a synchronized manner in order to satisfy a functional requirement (FR). The service constitutes a component that exposes its functionalities through interfaces under the specific language of the definition [1]. In combination with other services, these functionalities are those that satisfy specific requirements. The composition task depends on placing logically arranged services and involves aspects defined in its non-functional requirements (NFRs), which constitute quality of service (QoS) criteria to select WSs [2]. The FRs establish which services will be in the WSC, and the NRFs establish which QoS criteria those services will have to satisfy. FRs provide the identifiers and interfaces of the services and how they will exchange messages. Usually, WS deployment is performed under cloud providers, such that attention should be paid to the resources consumed by each of the distributed providers. Consequently, user interaction and resource consumption are essential factors

to consider during composition. The exponential growth in services with different QoS levels over providers [3] with the same functionality poses challenges in selecting a WS, as there may be services with the same functionalities in different providers, but with different QoS values [4].

The advantages of cloud environments can create inherent problems generated by vendors whose deployment of WSs on different nodes [5] can generate a variety of QoS criteria. Thus, non-functional requirements must be analyzed with caution, such as the ability to establish agreements for the provisioning and acquisition of computational resources (static or dynamic negotiations), if the composition requires it [6].

Each service exchanges messages with its peers through connectors of an atomic service (a simple WS). In turn, a complex or composite service (joining several WSs) is developed, which interacts through some functional logic. Its scalability and growth levels mainly depend on the flexibility of the selected services, such that, if a service is independent of constant QoS criteria, these can be used later by another composite service [6]. The web service selection problem has been described in [7], where the emphasis was placed on the exponential growth of services with different QoS for the same functionality, leading to combinatorial computational costs for the search and selection processes.

Our research constructed a computational strategy to discover and select WSs within the pattern-recognition domain. The strategy builds from modeling services and their characteristics as code units to be recognized by a pattern system, dynamically created by the service composition model. Pattern recognition offers methods and algorithms for the classification of objects based on the analysis of their attributes. Our classification approach models the system using the concept of learning—this refers to algorithms capable of solving the unknowns in the patterns. This solution allows for eliminating the noise in the information, increasing the reliability of its treatment.

Learning can occur in different ways, as has been reported in the literature. Within these approaches, Ar2p [8] uses supervised learning through a strategy that offers an overview of pattern modeling based on the systematic functioning of the brain. Our proposed method uses a different approach than Ar2p, at the level of the recognition strategy: the original Ar2p recognizes entities using an interval model to classify the pattern previously loaded in the system. On the other side, our approach dynamically analyzes the classification pattern and decomposes it for later analysis by the different recognizer modules, dynamically created given the user's requirements. Its process consists of decomposing each QoS parameter to be analyzed by dynamic recognition modules until the desired WS is achieved for the composition required by a user. Such processing analyzes the characteristics of the entities (patterns) to be assigned to a category type. The main contributions of this paper are as follows:

First, we establish a theoretical framework for the implemented strategy, which emphasizes pattern recognition through the specification of a supervised learning classifier. It uses a variable selection segmentation process based on the analysis of QoS parameters. The WS classifier creates a mechanism for the hierarchization or WS ranking. The mechanism calculates the service score according to the QoS parameters registered by the user for each service and the composite service. Thus, this ranking mechanism offers a range of quality solutions, where the user chooses the most appropriate services according to the quality they wish to obtain through their composition. Second, we develop a formal specification of web services at the atomic and composition levels, representing information entities representing the patterns and signals to be recognized. In this process, a simple approach is offered to decompose the problem of combinatorial nature into a search and selection strategy of services based on dynamic recognition modules. The latter feature is provided when the system can create modules from the requirements of a user. Finally, a computational strategy based on a component architecture is defined. The strategy independently and dynamically develops analysis modules for each non-functional requirement of the composition. A mechanism for the categorization of WSs using QoS levels is finally established. Our method bases its computational strategy on selecting the services

required for a composition, thus establishing different solution pathways (SPs) or sets of services involved in the composition. They are categorized with different metrics, offering the system a more comprehensive range of searching and selecting.

The organization of the remainder of this paper is as follows:

- Section 2 discusses research on the selection of web services related to the recognition approach.
- Section 3 outlines the formalization of the process of composition, web services, and QoS parameters, represented through signals and patterns by Ar_WSDS.
- The Ar_WSDS method is presented in Sections 4 and 5, specifying the recognition module concept and strategies, the system model, and the proposal of the algorithm solution.
- In Sections 6 and 7, the experimental phase is presented and discussed, using a data set to compare the proposed system to another recognition method with similar heuristics to analyze the developed algorithm's performance. The paper is concluded in Section 8.

2. Background and Related Work

2.1. Web Service Composition

Service composition-oriented systems are widely linked to cloud computing. Beyond analyzing the cloud paradigm as a deployment platform, a series of relevant performance and accessibility requirements is established when selecting services.

A cloud service and composition lookup proposal has been presented in [9], based on the Cuckoo lookup algorithm through constraint-satisfaction techniques. Its test scenarios were based on distributed cloud environments with WS replicas having different QoS values. This paper provided insight into the use of multi-clouds with their respective data sets, highlighting the use of filters for the parsing process and for the functional part of each candidate WS to be selected by the composition model to be satisfied. Likewise, in [10], a pattern-based selection method using a composition and ontology-level search has been presented. The WS formalization was highlighted, in this work, in terms of enabling the accurate description of its interfaces and parameters, circumstantially streamlining the WS selection process at the syntactic level. In [11], a method to discover and select services using machine learning has been proposed. The system predicts new QoS values from information found in its web service description language (WSDL). Its analysis was based on obtaining data directly from the WSDL, as it describes how to compose a web service request and describes the interface provided by the provider to consume it. The analysis allows for filtering those WSs that are available at a given time.

Research that developed the notion of pattern discovery and classifications in WS has been proposed in [12]. The authors used data mining techniques to classify services through specific non-functional requirements. Their study concluded that the main points of analysis in a WS are the response time, performance, and availability, which are critical metrics for a composition activity. In [13], a service discovery method based on Petri nets was used, developing a whole composition system modeled on a cluster of services in real-time. Its system provides the service with a semantic level, which allows for labeling it with information from the analysis of its QoS values, in order to create groupings of services according to a particular non-functional requirement.

In [14], an algorithm has been presented for WS selection in repositories containing a large number of services. Their computational strategy was based on the theory of equivalence relations, and they modeled a system for storing services that avoids redundancy in its registration semantics. Their experiments were based on sequential composition models, where each service matches or connects to its predecessor service. Qi et al. [4] have emphasized the composition problem in cloud environments at the WS selection time with the description of scenarios with replicas. An algorithm and mathematical model were developed using differential evolution-based knowledge-learning heuristics, based on searches on three QoS criteria: Response time, service availability, and reliability. The testing scenarios used random data to verify the performance of the algorithm. This aspect is

essential, as data repositories in this field are scarce, and many researchers have chosen to create their own data sets.

Hasnain et al. [15] proposed analyses on different data sets, presenting the main parameters considered for the selection of WS, including the response time, latency, and reliability. They also described the importance of conducting studies on QoS parameters which can impact the cloud environment.

Rathore et al. [16] have developed an algorithm called OAEQoS. Its important contribution consists of the analysis of web services by monitoring QoS parameters performed through the access of information to their providers, a concept which is important when considering the type of repository to choose for the search of services. Its evaluation adds a reliability parameter that consists of performing service invocations and analyzing their successful and unsuccessful requests. This process allows for updating parameters such as availability, throughput, successability, and reliability. The process only considers services that are active, increasing the level of reliability for the selection of a service.

The above-cited works provide an overview of the composition process in its selection stage, which uses QoS parameters as a selection filter. Thus, Ar_WSDS is characterized by a strategy based on the dynamic analysis of QoS parameters for composition models which can vary their quality strategy to discover and select WSs, and we adaptively devised its core system for new non-functional requirements of users.

2.2. Pattern Recognition

Pattern recognition methods have been extensively studied [17,18], where the related works have mainly targeted selection methods based on pre-defined features over a data set. The main goal of pattern recognition is to find and characterize structured information (called patterns) from information sources that present data in a dispersed manner. Different strategies allow for the identification of trends with the recognition of a specific descriptor (attribute) [19]. Two recognition strategies have been defined: Supervised and non-supervised. In the former, recognition is based on the presence of structured patterns. With this, the prediction of patterns of interest is carried out. At this stage, training is necessary, in order to determine the membership of the pattern to a specific class. In contrast, the latter strategy can be used to predict behaviors and relationships between the data, in order to determine the membership function of a class [20].

We focus on supervised strategies, where the construction of the predictive model has the objective of assigning patterns to classes that have already been defined [21]. In general, using these approaches requires pre-established knowledge to train the model. Such methods base their computational strategy on the elimination of redundant variables and on the fast recognition of an entity with active patterns or first-order rules, with which it must comply to be recognized [22]. Each characteristic of the entity represents a variable that needs to be analyzed during the recognition process. So, through any machine learning algorithm, the entity is categorized. In recent years, research has provided machine learning mechanisms that can analyze a variable set and whose computational cost is acceptable for a data set that, in basic processing, can have an NP-order cost [23].

One of these recognition models is Ar2p [8], which is based on the systematic functioning of the brain. This paper proposes its adaptation to develop a recognition model for WS selection in composition activities under cloud environments. Ar2p was originally developed from the pattern recognition theory of mind (PRTM), where the memory manages a hierarchy of patterns, which are perceived through our senses (conceptually called "sensors") [24]. As soon as one of our senses is activated, certain patterns can be recognized.

A general model of Ar2p is shown in Figure 1, where each level represents a recognition module. Thus, a system can have several levels which offer the possibility of initially recognizing atomic patterns (properties with a single attribute/value); however, a pattern may pass through several levels until it is recognized, resulting in a complex pattern that underlies the processing of the sub-levels. Every recognition module's target is to recognize its corresponding pattern (i.e., a copy of the pattern), such that recognition can occur at any

level. Ar2p defines two strategies to recognize a pattern: One by key signals and another by partial signal recognition. The former uses the importance weights of the input signals identified as key signals, while the latter uses the partial or total presence of the signals in a simplified model.

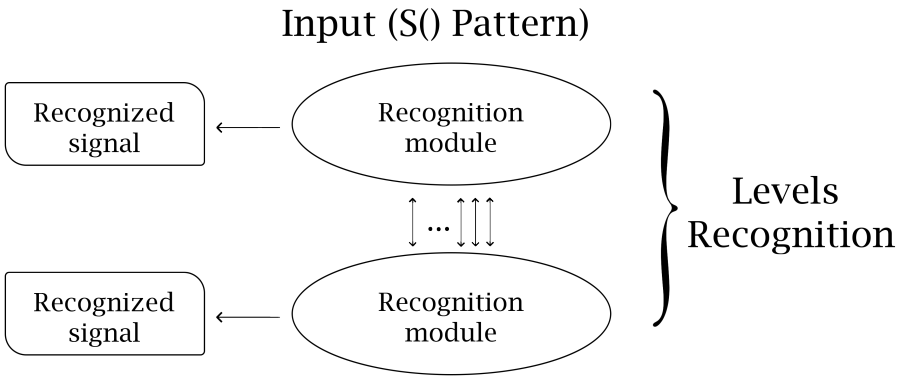

Figure 1. Ar2p model [8].

A pattern can be recognized by a single key signal or by the total of the partial signals if and only if they pass through the observation space based on their recognition levels. The advantage of using Ar2p is its level of uniformity, as it ensures that each recognition function is the same. This process is executed recursively. Considering PRTM [25], the recognition process occurs when the brain can recognize a pattern through any sense. Thus, the brain's ability sharpens when developing a hierarchy of complex patterns (i.e., a learning process occurs).

Below, we formalize the concept of composition and discuss how a composition is modeled to be used as input to an Ar_WSDS recognition process.

3. Abstract Composition Model

An abstract composition model (ACM) of WSs is a sequential workflow, as shown in Figure 2, represented through an acyclic graph of size n, composed of the WSs $1, \ldots, n$. Equation (1) shows WS_T, which lists the services (nodes) needed to fulfill the composition. Each WS_i consists of a 3-tuple: <I, O, [QoS]>, where I is its input parameters, O is its output interface, and $[QoS]$ is its quality parameter matrix.

$$WS_T = \{WS_i \mid WS_i \in ACM\}. \quad (1)$$

As such, there exists a matrix, QoS, such that $QoS_{i,j}$ is the value of a quality attribute j for a WS at position i within the ACM; that is, WS_1 has a value $QoS_{1,0}$, which can be, for example, a latency value and so on for each service. Therefore, each WS has an associated QoS parameter matrix. The set of these is indicated by an array of QoS_T matrixes.

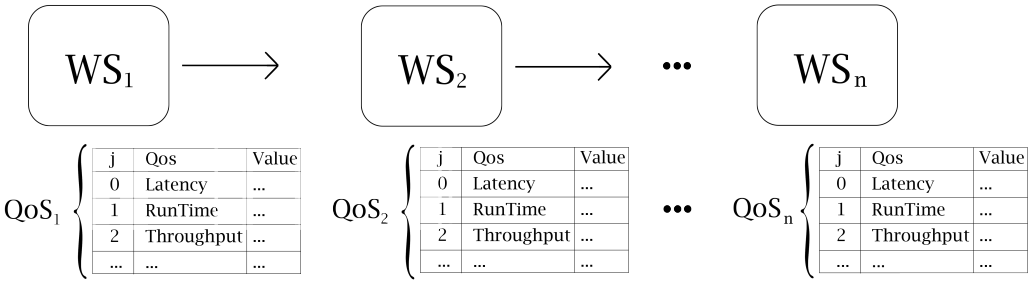

Figure 2. Abstract composition model for the Ar_WSDS approach.

Every step in the ACM requires the use of one WS_i. The composition goal is achieved when all of the WS_i are available and correlated in the model, such that the execution is based on the ordered path of each node in fulfilling the adjacent functionality. Although other types of workflow models may exist, they may be transformed into sequential models [26]. All WSs must match the assembly condition provided by its composition model, which corresponds to the request and invocation of a predecessor WS with its predecessor candidate. Figure 3 shows the transitive relationship between the parameters I and O, with respect to each other.

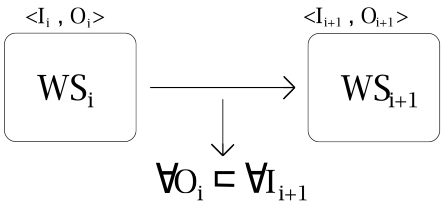

Figure 3. Web service assembly structure.

Web services input to the system are denoted as the set WS_g, where WS_{g_i} is a general web service to be recognized by the system. A candidate WS, denoted by WS_c, is a service that has been discovered and selected from the WS_g, and is considered a possible solution for a node of the composition model. Combining the WS_c creates the SP, or different subsets that satisfy the ACM and the QoS conditions provided by the user for composition.

The following section presents the Ar_WSDS design, specified through the presented ACM model.

4. Overall Design of Ar_WSDS

The Ar_WSDS architecture shown in Figure 4 includes eight steps. First, the input of the system, whose function is to read the data and create the WS_g set, corresponds to the input of the system.

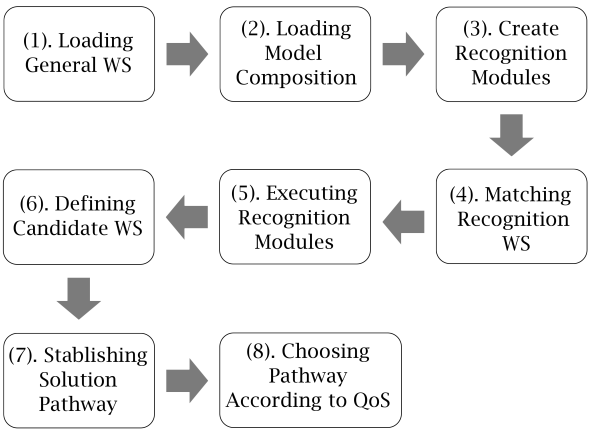

Figure 4. Ar_WSDS workflow.

Second, loading the ACM defines the service syntax requirements and the QoS parameters at the atomic and composed level (resulting service after composition). Third, it defines the dynamic creation of the recognition modules that conform to the ACM, through the two strategies set up in Ar_WSDS (as discussed in the next section). Fourth, matching by service name is performed, allowing the system to reduce its search space and limit its analysis to QoS. Fifth is the execution of the recognition modules, which enables the selection of the most appropriate services according to the user's QoS requirements.

Sixth, the candidate services are set up. Finally, in the seventh and eighth stages, the SP establishes the set of services that satisfy each node and whose union maximizes the user's quality level, defined as the value of their composed service.

Below, the system definitions and the semantic notations used in Ar_WSDS, as a series of components that identify the system within the PRTM environment, are provided.

Definition 1. *A pattern (ρ) is an entity represented by a WS_{g_i} to be recognized through the parameters within the QoS_T set. Based on the decomposition of ρ, the set of atomic signals, named $S()$, is generated.*

$S()$ contains information about each WS_{g_i}, and the recognition modules classify the service as a candidate if it complies with the assembly and QoS characteristics.

Definition 2. *An optimal WS is the service that is closest to its QoS parameter specified in the composition model.*

SP is a subset of optimal web services, and the system recognizes 0 or n pathways. If its value is 0, it means that some service did not comply with its respective QoS parameter. Therefore, an ACM node will not have any associated services. Figure 5 shows an example of the Ar_WSDS system, where the inputs are the specifications WS_g and the ACM. Then, the recognition modules discover and select the most suitable WS for the model node. Finally, the system creates the SP, in order to satisfy the requirements provided in the ACM.

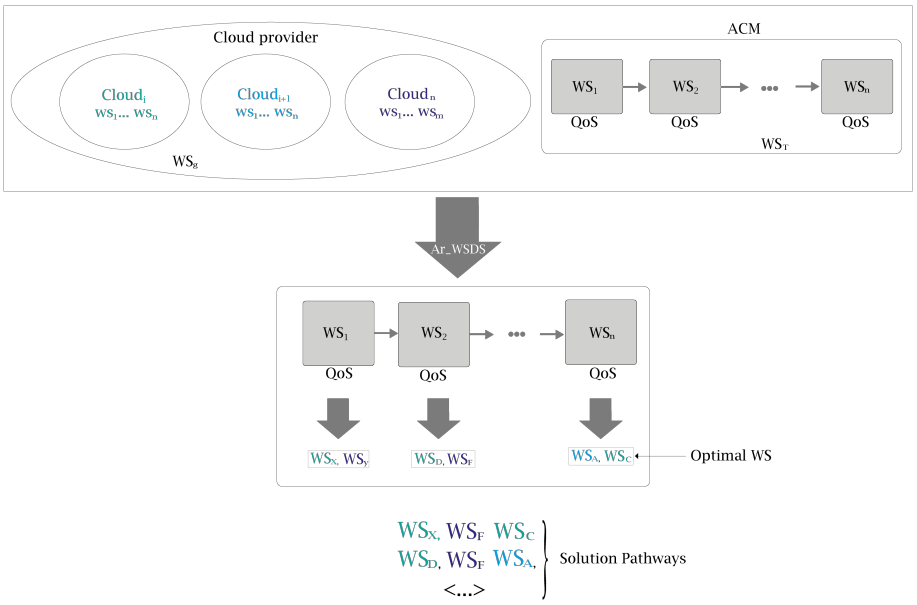

Figure 5. Example of Ar_WSDS operation.

Definition 3. *A recognition module, symbolized as Γ, consists of a set of software components that recognize a pattern and its signals.*

Due to the modularity and architecture of the Ar_WSDS, these modules are dynamic (i.e., automatically adaptable to the parameters of each WS_i of the composition model). Equation (2) formalizes the recognition modules:

$$\Gamma = \{\Gamma_{i,j} \mid \Gamma_{i,j} \in QoS_{i,j}\}, \quad (2)$$

where

1. $\Gamma_{i,j}$ is a recognition module for service i of its QoS parameter j, and
2. $QoS_{i,j}$ corresponds to QoS parameter value j of service i.

Definition 4. *A WS ranking, symbolized as φ, is a value of the degree or weight of importance of a web service candidate once the Γ modules recognize it.*

Hence, $\varphi_{i,j}$ has a value of 1 when the value for $QoS_{i,j}$ of the WS_{g_i} is categorized as the most optimal, and a value of n when it is further away from the optimal WS.

Definition 5. *The matching recognition function, represented as $M()$, evaluates whether the name of the service described in set WS_T has a matching correspondence in the set WS_g, in which case, its result is true, and it is false otherwise.*

$$\exists_x \{x \mid x \in WS_c \leftrightarrow M(WS_g) = true\}. \quad (3)$$

Equation (3) shows the function $M()$, which aims to minimize the search space between the WS_g and to create the primary candidates (elements detonated as x). These are WSs to be treated by the recognition modules, which analyze each value of their QoS.

Definition 6. *The quality thresholds, represented as Δ, are discrete values that must be met by the composition (composed service) and each WS_{g_i} (atomic service) to be a candidate service.*

The value of Δ depends on the QoS parameter; for these values, its optimum will be given as a maximum or minimum. The characterization of attributes has been described in [27], and analyzed using a data set approach in [15]. There are three different types of Δ:

1. ΔWSC is a value (percentage) that represents the quality condition that must be met by the composed service through the process of analyzing the WS_c and its respective φ, in order to be considered within an SP.
2. A key threshold (ΔKS) is a value of a quality parameter for a WS_{g_i} that must be in the acceptance interval of its respective $WS_{g_{i,j}}$. As previously mentioned, it depends on the nature of the parameter and can be expressed as a minimum or maximum. The delta must be fulfilled.
3. A partial threshold (ΔPS) is a quality parameter value for a WS_{g_i} which is above or below the average of its QoS of the same type. These values may be below the quality condition. The φ_{ij} value is subjected to a penalty function, denoted by $\beta(\varphi)$, which presents the ranking of the $QoS_{i,j}$ value and whose ranking values are calculated from the difference between the best service and the service furthest away from the quality condition.

Thus, for a service to be considered a candidate, it must meet the thresholds defined in ΔKS or ΔPS.

Definition 7. *A recognition type is a value between 0 and 1 associated with a $QoS_{i,j}$ to use a Δ; if the value is 1, it will use ΔKS, while 0 indicates the use of ΔPS. No Δ value means the QoS parameter will not be considered by the recognition modules.*

In Figure 2, a service is extracted, while Figure 6 presents an example of a service specification of the WS_T set; the values and type are presented. The latency and throughput value are processed by key thresholds, at runtime, using a partial threshold (average value).

The following section defines the recognition strategies developed in Ar_WSDS, based on the definitions mentioned above.

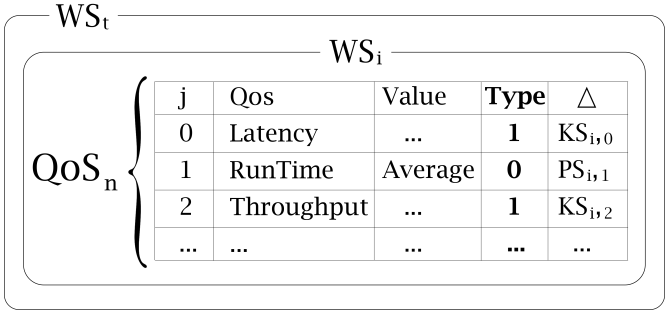

Figure 6. WS specification and recognition type.

5. Recognition Strategies

The system uses two strategies for the recognition of the set of signals from the thresholds KS and PS, in which they are decomposed into atomic signals $S()$ corresponding to the $QoS_{i,j}$ constraints, formalized in the following definitions.

Definition 8. *Module recognition by key signals is a recognition process that is enabled when the $\Gamma_{i,j}$ module analyzes ΔKS for each WS_{g_i} and obtains $\varphi_{i,j} \in \{1..n\}$. If $\varphi_{i,j} = 0$, the S() signal is not triggered, and the analyzed element of the WS_g set is discarded from WS_c.*

Equation (4) models the recognition of a WS_{g_i} pattern, when the sum of all its rankings generated from the $\Gamma_{i,j}$ evaluation in its $QoS_{i,j}$ is greater than or equal to its corresponding $\Delta KS_{i,j}$. The numerator of the equation at summation determines the exact proportionality of QoS for the service to be recognized. The denominator determines the total contribution of that parameter to achieve the ΔKS of the analyzed service.

$$KS\_Recognizer(WS_{g_i}) = \sum_{j=1}^{|QoS_i|} \left(\frac{\frac{100}{|QoS_i|}}{\varphi_{i,j}} \right) \geq \Delta KS_{i,j} \rightarrow WS_{g_i} \in WS_c. \quad (4)$$

Definition 9. *Module recognition by partial signals is a recognition process, which is enabled when the $\Gamma_{i,j}$ module analyzes each WS_{g_i} and its ΔPS (which is obtained as an average of the QoS of the service of the identical type), and obtains $\varphi_{i,j} \in \{1..n\}$. If $\varphi_{i,j} = 0$, the $S()$ signal is not triggered, and the analyzed element of the WS_g set is discarded from WS_c.*

$$PS\_Recognizer(WS_{g_i}) = \sum_{j=1}^{|QoS_i|} \left(\frac{\frac{100}{|QoS_i|}}{\beta(\varphi_{i,j})} \right) \geq \Delta PS_{i,j} \rightarrow WS_{g_i} \in WS_c. \quad (5)$$

A partial signal activates a partial signal recognition module when the sum of all its rankings generated from $\Gamma_{i,j}$ in the β penalty function is greater than or equal to the $\beta(\varphi)$ of the service to be recognized, as formalized in Equation (5). The penalty function allows us to find a value close to 1 when the QoS parameter of the WS_g is at the level of the average and substantial values for parameters far from the average. Thus, it offers the possibility for the system to rank the WSs from the most to the least appropriate, in terms of reaching the respective ΔPS.

The searching and selecting of the WS_c is performed through the recognition modules. The system begins when the $S()$ signals containing the information of the atomic WSs are recognized by the $M()$ function. Its objective is to syntactically verify whether the WS corresponds to the identification requested by the ACM. Then, the signals are analyzed according to their nature and their KS or PS recognition type, the rankings generated by the different recognition modules are calculated, and a list of the WS_c that guarantee ΔWSC is generated.

Figure 7 depicts the Ar_WSDS recognition system, which receives input from its composition model and the web services to be classified from a repository. The $M()$

function chooses the primary candidates and sends them to the respective recognition module, which depends on the type of recognition and the QoS type. The system processes and generates the candidates to create the solution pathways that depend on ΔWSC.

Based on Figure 7, Algorithm 1 was developed, which specifies the Ar_WSDS procedure, as detailed below:

1. Read data from the WS_g set.
2. Read data from the composition model. Using this procedure, it is possible to specify services and their QoS attributes for each atomic service and its composed service (Δ specifications). Information is loaded into the WS_T array.
3. Creation of candidate matrix ($MC[][]$). This matrix has several rows, depending on the cardinality of the WS_T array. The columns depend on the number of services recognized for that particular node.
4. Creation of the primary candidates, filtered by the matching recognition function $M()$; creating the initial WS_{CT} set.
5. For each WS_i, WS_T is decomposed into sub-patterns $\rho(WS_i)$. They generate the corresponding recognizer modules Γ, which are responsible for the core of the recognition process from the QoS parameters and ΔKS, or by ΔPS, according to the user's preferences. For each WS_c, its recognition module Γ is called and its ranking (φ) is calculated.
6. The recognition function selects the candidates from the calculation formalized in Equations (4) and (5). When the recognition is valid, WS_c is stored in the MC matrix. Equation (6) formally defines the recognition element, where n is the number of $WS \in WS_T$ and m is a value representing the number of WSs that satisfies an ACM node. The element x, defined in Equation (6), is a WS used for solution pathways.

$$MC_{nxm} = \exists_x \left\{ x \mid x \in \begin{bmatrix} WS_{c_{1,1}} & \dots & WS_{c_{1,m}} \\ WS_{c_{2,1}} & \dots & WS_{c_{2,m}} \\ \dots & \dots & \dots \\ WS_{c_{n,1}} & \dots & WS_{c_{n,m}} \end{bmatrix} \leftrightarrow (KS(x) \bigoplus PS(x)) \right\}. \quad (6)$$

7. The process is not valid if a WS_c does not exist for any WS_i.
8. The $I(M_c)$ function is responsible for integrating the solution pathways defined in MC, a process conducted by grouping the candidate services for each WS of the model and selecting them, according to the condition ΔWSC that satisfies the composition model.

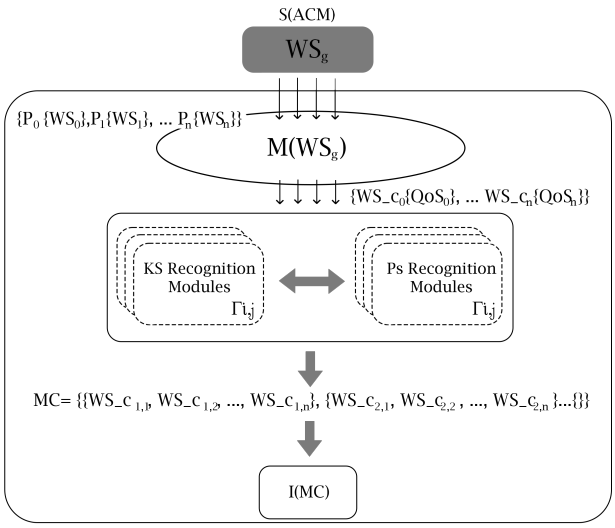

Figure 7. Ar_WSDS recognition system.

Algorithm 1: Pattern recognition algorithm of web services.

Input: ACM,WS_g[]
Result: MC[][]
WS_T[] = load(ACM);
WS_{CT}[] = M(WS_g);
MC[0..|WS_T| - 1] [ ] *be_a_new_matrix*;
$i = 0$;
for $WS_i \in W_T$ **do**
  $\Gamma = \rho(WS_i)$;
  $j = 0$;
  for $WS_c \in W_{CT}$ **do**
    if $\Gamma(WS_c)$ **and** $\Gamma \in KS$ **then**
      isC = KS_Recognizer(WS_c);
    else
      isC = PS_Recognizer(WS_c);
    end
    if $isC = true$ **then**
      $MC[i][j]$ = WS_c;
      $j = j + 1$;
    else
      delete WS_c in WS_{CT};
    end
    if $M[i] = \emptyset$ **then**
      Error no candidate services;
      exit;
    end
    $i = i + 1$;
  end
  Create solution pathways based on MC;
  $I(MC)$;
end

6. Evaluation and Results

We tested Ar_WSDS with the WSs in the QWS Data set project [28], with a replica data set to increase the search space. According to their features, values were analyzed and placed into two groups, corresponding to values that should be maximized or minimized, based on the analysis in [29]. As Ar_WSDS test scenarios, we selected several WSs in sequential workflows to analyze the proposed approach's effectiveness and performance. Figure 8 shows an example model of a basic scenario with a workflow considering four WSs with KS recognition and a ΔWSC value. In the solution, we can see that two services satisfied WS_1, one service satisfies WS_2, two for WS_3, and two for WS_4. The WSs were stored in the MC.

Using the function $I(MC)$, the solution pathways greater than or equal to the value defined in ΔWSC were created. Hence, this quality condition was inversely proportional to the number of tuples of the solution pathways.

6.1. Experimental Setup

We dynamically designed the experiments using a Python application, loading the data set with the addition of replicate data according to the type of the original data set. Thus, it was possible to operate with multiple cardinalities for the set WS_g. The experiments consisted of two groups, depending on the used strategy, either KS or PS recognition. Each one considered a variable number of active QoS (signals) and several WSs (WSs in ACM). Solution pathways for each WS depended on the number of WS candidates for each WS, described by the matrix defined in Equation (6). The experiment setup used the same

hardware platform, running on a Digital Ocean virtualized server with 4 GB RAM, 1 CPU, the Linux Ubuntu 18.04.3 LTS operating system, and with Python version 3.6.

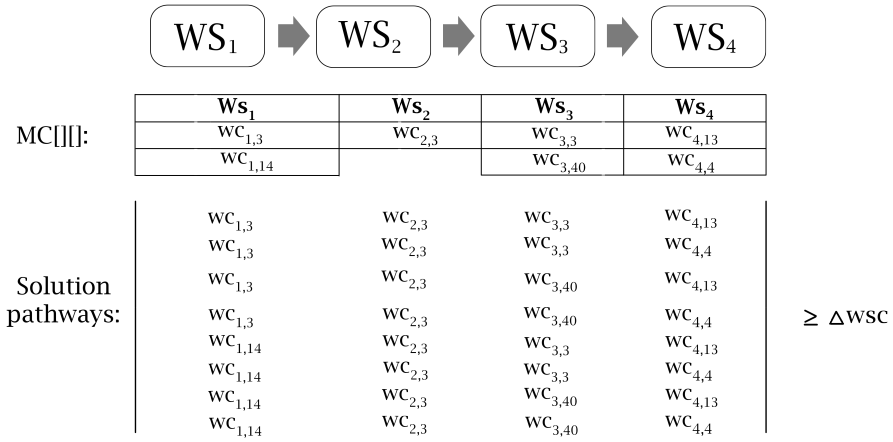

Figure 8. A WS recognition case study.

There were two frontends to the application: The first was a console allowing for the massive loading of services, while the second was a web view to be managed by the user, through options that guided them in the service discovery from its specific requirements. Outputs were presented in a plain text file, using a table easily exported to a spreadsheet or pdf format.

6.2. Experiments' Parameters

We chose 100 services with replicas to force the algorithm to perform comprehensive searches and selection. The replication rate of each WS_i varied between 20 and 50 over different clouds. Table 1 describes the QoS and its features, allowing each recognition module to define its value analysis strategy based on minimum or maximum data intervals. Each QoS parameter was set with a P_1 to P_9 identifier. For instance, for P_1 the analysis of its quality values was performed using minimums, which enabled each QoS parameter to be evaluated according to its character.

The experiments ran five scenarios, each recognizing WSs based on a value of ΔWSC, ΔKS, or ΔPS. The first scenario evaluated key signal recognition, and the second scenario evaluated partial signal recognition. Besides that, the third scenario combined key and partial recognition. The fourth scenario developed a case study of a real-time composition environment, and finally, the last scenario simulated the WSs ranking.

The first and second scenarios used three data sets, with 10, 50, and 100 WSs having with two, four, and nine signals, respectively. A set of two signals used P_1 and P_2, four signals used P_1 to P_4, and nine signals used P_1 to P_9 QoS.

The third scenario used 50 WSs, and each test combines the two types of recognition. We defined a series of tests to combine P_1 to P_9 QoS. The system started its recognition using relations of 2 to 7, 4 to 5, and 1 to 8 partial and key signals, respectively. Other tests inverted these relations so that the system recognized using key signals first. The fourth scenario used three WSs of an ACM.

Finally, the fifth scenario evaluated Ar_WSDS concerning other approaches. Tests were performed on a set of nine services to be selected, and all of them had replicas. The test was performed using the following initial conditions for Ar_WSDS:

- Reliability, availability, and response time were the only signals activated;
- $\Delta WSC = 90\%$ was defined such that the best WS must be chosen; and
- Key signal recognition used WS values that were very close to their boundary values, which enabled the system to select services with high quality.

We will now show the results of each scenario.

Table 1. QoS parameter description for WS selection.

Parameter Name	Description	Units	Max (+)/Min (−)
Response Time (P_1)	Time taken to send a request and to receive the response.	ms	−
Availability (P_2)	Time period in which a web service exists or is ready for use.	%	+
Throughput (P_3)	Number of services that a platform providing web services can process for a unit time.	invokes/second	+
Successability (P_4)	Message success ratio after invocation of a service.	%	+
Reliability (P_5)	Rate of failures over a period of time	%	−
Compliance (P_6)	WSDL compliance rate based on its formal W3C specification.	%	+
Latency (P_7)	Delay time to process a request.	%	−
Best Practices (P_8)	Service specification compliance ratio using WS-I Basic Profile.	%	+
Documentation (P_9)	Compliance ratio the WSDL.	%	+

Adapted from [28].

6.3. KS Recognition Scenarios

KS recognition defined the search and selection in a restrictive manner. Therefore, the higher the number of active signals, the lower the cardinality of the search. Dynamically created KS (Γ) recognition modules trigger their operation, according to the user-defined value of their respective ΔKS. These values were chosen in such a configuration that the most significant number of WS could be recognized.

Table 2 presents the results of the key signal evaluation. The WS column represents the number of WS_c, while KS represents the number of signals. The WS_T column represents the number of services in ACM. The runtime was measured in seconds for console outputs. Detail is the time set after reading the input data, and the SP column corresponds to the number of n-tuples under $\Delta WSC \geq 80$.

Table 2. Runtime results using KS.

WS_T	KS	y	Run Time (s)	SP
10	2	17	0.5	8
50	2	129	5.82	34
100	2	1047	122.3	166
10	4	10	0.3	1
50	4	81	3.21	21
100	4	421	39	58
10	9	10	0.17	1
50	9	50	2.34	1
100	9	100	6.02	1

6.4. PS Recognition Scenarios

We used the exact scenario definition of KS recognition, as this type of recognition had a more significant WS; therefore, the number of solution pathways may increase. The $\beta(\varphi)$ function depended on the average value for each $QoS_{i,j}$ to be recognized. As such, the processing required by Ar_WSDS was considerably increased.

Table 3 presents the results of the partial signal recognition phase, showing how the number of candidate services increases. This condition arose as some WSs could

have high or equal values in their QoS, generating a higher number of WSs belonging to that interval of their average value, which is a different result from that considering the limitation imposed by KS recognition. Moreover, the results showed a tendency to find fewer solution paths when all nine signals were active for recognition. For very high values of candidate WSs, the processing time to find the routes increased. The measured $\beta(\varphi)$ was one and two, in order to increase the coverage of the solution space.

Table 3. Runtime results using PS.

WS_T	PS	WS_C	Run Time (s)	SP
10	2	180	2.13	42
50	2	418	3.25	56
100	2	5620	427.2	402
10	4	83	3.07	28
50	4	137	4.23	39
100	4	1827	201.3	193
10	9	32	1.2	13
50	9	63	4.22	19
100	9	731	71.2	67

6.5. KS and PS Recognition Scenarios

The Ar_WSDS system defined its recognition model based on the discovery of the patterns defined in Figure 7 and Equation (6), where it is established that, if the key signals could not recognize a pattern, they were recognized by partial signals. Success rate tests for each of the recognitions in the nine QoS parameters were conducted, and Table 4 presents the results.

For each experiment, the number of signals to be recognized was nine (i.e., there was a variation between partial and key signals), such that the system could recognize one group of signals if it did not recognize the other. All recognition modules were dynamically created in this scenario. This setup means that, as the constraints increased in both recognitions, the recognition success rate decreased. In practice, this condition implies that, if a service with high restrictions in its non-functional requirements is requested, few services will satisfy this quality condition.

Table 4. PS and KS recognition success rate results.

PS	KS	PS Success Rate	KS Success Rate
2	7	82	18
4	5	67	33
1	8	91	9
7	2	21	79
5	4	41	59
8	1	11	89

6.6. Case Study: Web Service Recognition by Key and Partial Signals

We assumed that the input patterns corresponded to the ACM shown in Figure 7. Each WS was recognized by the four key and five partial signals, as described in Table 5. Each WS discerned exactly when to execute its operations and with whom it had to interact [30]. Its main objective was the collaborative exchange of messages between the WS in public business processes.

Figure 9 depicts an example of a composite service for storing messages on a file server, showing the interaction of the DOTSGeoPhone, BINDService, and WebStoreService services, this comprises their FRs. The user requirement threshold for the composite service was 90%. Table 5 presents the WS after the $M()$ function was invoked, and the recognition features are shown in Table 6.

The recognition strategy set the first four QoSs to be recognized by key signals and the rest as partials. Table 6 presents the ΔKS values for each service. The ΔPS was calculated from their average, as exemplified in Figure 6.

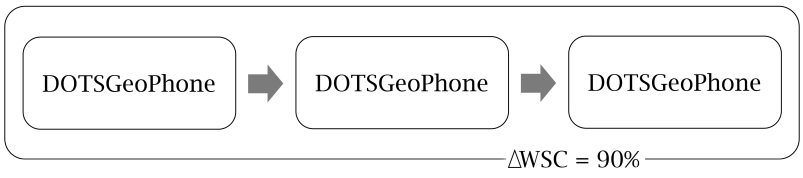

Figure 9. ACM recognition case study.

Table 5. Case study web services feature description.

Service Name	1	2	3	4	5	6	7	8	9
BINDService	266.83 261	36 36	0.9 0.9	37 37	60 60	89 89	69 69	15.91 13.67	7 12
DOTSGeoPhone	102 107.4	90 85	18.6 19	97 95	73 73	78 78	80 80	1 0.8	92 92
WebStoreService	251.07	56	52	58	73	67	80	133.43	31

(1) Response time, (2) availability, (3) throughput, (4) success, (5) reliability, (6) compliance, (7) best practices, (8) latency, and (9) documentation.

Table 6. Case study web services description of the type of recognition performed.

Service Name	1	2	3	4	5	6	7	8	9
BINDService	KS = 300	KS = 36	KS = 0.8	KS = 36	PS	PS	PS	PS	PS
DOTSGeoPhone	KS = 106	KS = 90	KS = 187	KS = 100	PS	PS	PS	PS	PS
WebStoreService	KS = 300	KS = 60	KS = 60	KS = 60	PS	PS	PS	PS	PS

(1) Response Time, (2) Availability, (3) Throughput, (4) Success, (5) Reliability, (6) Compliance, (7) Best Practices, (8) Latency and (9)Documentation.

Based on recognition by the function $M()$ and Equations (4) and (5), the method worked as follows:

1. The QoS matrix was derived from Equation (2). It allowed for defining the values to be used by the recognition modules (Γ).

$$QoS = \begin{bmatrix} 266.83 & 36 & 0.9 & 37 & 60 & 89 & 69 & 15.91 & 7 \\ 261 & 36 & 0.9 & 37 & 60 & 89 & 69 & 13.67 & 12 \\ 102 & 90 & 18.6 & 97 & 73 & 78 & 80 & 1 & 92 \\ 107.4 & 85 & 19 & 95 & 73 & 78 & 80 & 0.8 & 92 \\ 251.07 & 56 & 52 & 58 & 73 & 67 & 80 & 133.43 & 31 \end{bmatrix}$$

2. Figure A1 shows the two BINDServices which were recognized as two candidates. Their four signals were active and with a respective ranking. However, the service with the best score was listed first. This process resulted in a signal from service 2 (response time = $QoS_{1,0} = 261$), which had a response time value which was more appropriate for the requested quality criteria.
3. Figure A2 shows a single service with partial recognition, which allows the system to easily and quickly choose the WS. Partial signals were active, and the service had the highest-ranking value. In this case, the first DOTSGeoPhone service was discarded, as its value of $QoS_{2,8} = 1$ was outside the average range. Key signals outside the Δ value did not recognize the services, and the system performed recognition using partial signals.

4. Figure A3 shows the recognition of the WebStoreService. The service was recognized as unique by the $M()$ function. The system analyzed all its signals partially, not by key signals, as the $QoS_{4,3}$ value did not satisfy its ΔKS value.
5. The $MC[][]$ matrix was generated and the SPs were processed, which specified the services that satisfy $\Delta WSC = 90\%$. Figure 10 shows the SP for the user's requirement, with each service displaying the information regarding its provider, its WSDL specification, and the ranking of the service. This allows the user to analyze different options, in order to choose services according to the required quality criteria.

Figure 10. Solution pathways of the case study.

6.7. Ar_WSDS Test with Ranking and Selection Algorithms

Ar_WSDS was compared to OAEQoS [16] and a generic discovery and selection algorithm (GDSA). The first algorithm was a simple method for calculating and analyzing QoS parameters, with the additional factor of checking the access and performance of the service; in the second one, a trivial solution through an exhaustive search was obtained. It consisted of analyzing each service to choose the candidates for the solution to the problem. This process meant that each possible solution was checked with the other ones. OAEQoS was customized such that its input and output parameters generated the list of candidate web services.

The frameworks for OAEQoS and GDSA are shown in Algorithms 2 and 3, respectively. Each of the algorithms used similar inputs to Ar_WSDS. However, the ACM was decomposed into more specific data, in order to load the required WS list and its QoS.

Algorithm 2: OAEQoS algorithm (adapted from [16]).

```
Input: service_list,service_list_required, and QoS_list_required
Result: service_list_candidate with Qos and rank
Read service_list,service_list_required, and QoS_list_required;
Select WS from service_list, considering service_list_required;
for s ∈ service_list do
    if s invoked process = true then
        Evaluate invoked failed and bounced;
        Update reliability, availability, and response_time;
        Calculate OAEQoS value;
        if OAEQoS is within the expected QoS_list then
            Calculate rank with the value of OAEQoS;
            Append s and its rank to service_list_candidate;
        end
    else
        s is rejected as a candidate;
    end
end
```

The input QWS Data set (*service_list*) was used. The reliability, availability, and response_time parameters were analyzed.

Test 1 was executed using key signal recognition, while test 2 was executed using partial signals. Table 7 shows the WS names used in the test and their corresponding IDs. As already mentioned, these services had replicas that created a significant discovery space.

Algorithm 3: Generic discovery and selection algorithm.

Input: *service_list*,*service_list_required*, and *QoS_list_required*
Result: *service_list_candidate* with Qos and rank
Read *service_list*,*service_list_required*, and *QoS_list_required*;
Select WS from *service_list*, considering *service_list_required*;
for $s \in$ *service_list* **do**
  Calculate average reliability, availability, and response_time for *s*;
  Calculate expected average from *QoS_list_required*;
  if *average is within the expected average* **then**
    Append *s* and its average to *service_list_candidate*;
  else
    s is rejected as a candidate;
  end
end
Sort *service_list_candidate* for each candidate service by its average value;
Rank based on the candidate's position in *service_list_candidate*;

Table 7. Descriptions of the WSs used for testing.

WS ID	WS Name	WS Quantity
1	GoogleSearchService	43
2	MathService	9
3	AWSECommerceService	25
4	OnlineMessenger	5
5	DownloadService	17
6	LandmarkService	8
7	WSDLInteropTestDocLitService	13
8	AmazonSearchService	13
9	TimeService	4

Tables 8 and 9 show the selection process evaluation concerning the expected services (effective WSs). The data in Table 8 only include the services recognized by analyzing their QoS parameters and determining which complied with the quality constraints. Table 9 shows the number of errors in the selection process. This result did not provide an analysis of the order, according to the rank of the service; it only analyzed the quantity and relevance of the service with the predicted WSs.

Table 8. Test results of the WS selection.

WS ID	Ar_WSDS Key	Ar_WSDS Partial	OAEQoS	GDSA	WS Effective
1	4	5	2	20	4
2	1	1	1	5	1
3	6	6	4	19	6
4	1	1	2	3	1
5	3	4	2	8	4
6	2	2	1	5	2
7	5	5	5	8	5
8	3	3	1	8	3
9	1	1	2	2	1

Table 9. Amount of errors in the selection of services.

WS ID	Ar_WSDS Key	Ar_WSDS Partial	OAEQoS	GDSA
1	0	1	2	16
2	0	0	0	4
3	0	0	2	13
4	0	0	1	2
5	1	0	2	4
6	0	0	1	3
7	0	0	0	3
8	0	0	2	5
9	0	0	1	1

Table A1 shows the expected services for each WS. The order of appearance corresponds to their rank, which determines the ranking position among the selected candidates; for example, WS_1 corresponds to WS ID = 1 which had good services, GoogleSearchService, which was in positions 3, 15, 28, and 31. Its results are presented as a two-tuple (WS ID, ranking), rankings ranged from 1 to n; if the ranking value is 0, it indicates that the service was not recognized among the candidates by Ar_WSDS. The results are discussed and analyzed in the following section.

7. Discussion

WS discovery and selection in cloud environments has generated interest in complying with quality requirements, thus offering users high-performance services and better access to providers. We tested our approach to obtain an overview of the behavior of services and the dynamic change in their QoS parameters, which differed significantly from the provider in which it is deployed. The redundancy inserted in the data set showed the differences between the services discovered and those selected. Thus, the execution time of the Ar_WSDS computational strategy increased as a function of the number of replicated WSs. However, under a natural environment, these scenarios can identify services with updated information having marked differences between their QoS, where the recognition provided by Ar_WSDS can result in considerable execution time improvements.

We consider the ACM as a problem in a sequential workflow; however, there are several types of workflows: selection workflow, parallel workflow, and cycle workflow; these depend mainly on a global QoS (QoS parameters that the composite service must have), these workflows through techniques can be converted to sequential workflows, processes that can generate more than one workflow for an ACM. The majority of solutions reported in the literature use QoS global [2,6,16,31], and how services contribute to achieving the objective, our system instead evaluates each service at the level of personalization of its QoS. The QoS global is taken as a quality value to be achieved by each solution path (set of candidate services for each node of the composition) created by our system. This type of adaptation allows for improved monitoring of the NFR of individual services. This feature is denoted in dynamic compositions that require structural changes in the functionality of a composite service without altering the FR for which it was created.

Figure 11 compares the recognition by key and partial signals using the average of the solution paths found for two, four, and nine active signals. Consequently, when the number of signals increased, the number of SP decreased. This situation occurred as the value of paths discovered by key signals was lower because this type of recognition is more restrictive and depends on the process executed in the Γ module, according to the QoS type. As the QoS parameters increase for selection, the discovery space shrinks. Thus, finding the SP for the input composition model will be performed with efficient runtimes, compared to solutions with sequential searches or detailed analyses for each QoS.

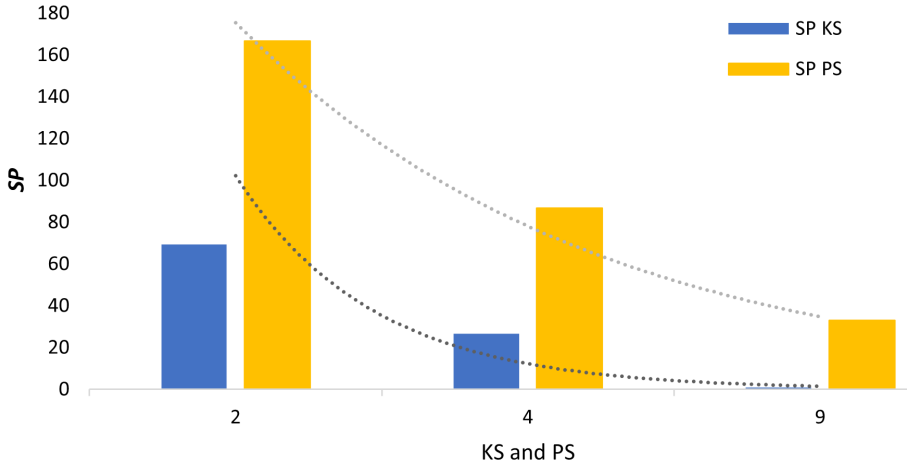

Figure 11. KS and PS signal recognition comparison.

The QoS parameters may vary; however, there are cases where the similarity of values makes choosing a service complicated. Ar_WSDS, using its recognition modules, can analyze all the parameters and find the differences between their values, thus allowing for the differentiation of a service from others. Table 5 displays this situation, where the similarity of QoS parameters for BINDService and DOTSGoePhone services are described using only three parameters.

Tests performed in the case study demonstrated the effectiveness of the recognition process, in its two stages, by key or partial signals. Table 6 shows how it is necessary to specify the thresholds for key recognition, which means that the programmer must know the desired QoS parameters in advance; however, partial recognition offers the advantage of analyzing all WSs and obtaining the most appropriate ones, providing a more comprehensive range of SPs for selecting services. Additionally, selecting a threshold for the composite service may cause the results to vary, according to their solution paths (i.e., the higher the threshold, the more restrictive the system is in selecting its services). Figure 10 shows how Ar_WSDS can adapt either of its two recognition strategies to select services, with partial signal recognition offering the best results, as mentioned above.

In the case study, WSs with a variable number of replicates were analyzed using two selection approaches—the first defined by OAEQoS and the second by GDSA—and both algorithms used input features from our approach. WS composition was performed sequentially from WS_1 to WS_9, allowing the system to have flexibility in selecting the service for each node on the ACM.

Tests were performed on the expected (effective) services, and our approach was compared with two existing approaches. Table 8 shows how Ar_WSDS achieved the expected services in its two recognition strategies, compared to the other two approaches; however, there were two cases where our approach recognized another service. This situation arose because the system analyzed all the selection possibilities, and the result of the comparison was performed based on the user's experience, such that the effective WS results may be different.

Figure 12 demonstrates the effectiveness of our approach, with GDSA producing more errors in the selection process as its solution strategy performed generic searches without considering any improvement idea that offered acceptable results to the user. Therefore, it is essential for the service selection process to have solution strategies that explore the deep analysis of QoS parameters considering their type. The system must analyze them in comparison to all the services that are similar in their functionality. OAEQoS produced acceptable results, and its capacity to retain updated QoS parameters was notable. However, the implementation of this approach necessitates the incorporation of additional routines to analyze more QoS parameters. Likewise, the update process was performed in remote invocations, which can improve the algorithm's performance, and its analysis became

static to the extent that the system was only based on pre-defined QoS parameters and only defined a selection strategy. In contrast, Ar_WSDS, which can dynamically create recognition modules for any QoS parameter and evaluate their thresholds on the basis of reaching the QoS threshold of the composite service, allows the user to obtain better results for each service involved in the composition.

Table A1 shows the ranking results of the WSs. We can appreciate that the two strategies of Ar_WSDS offer a more appropriate selection, compared to the other two approaches. A slight ranking difference existed in key vs. partial recognition, as the latter had a more extensive search space. OAEQoS presented differences under a larger number of WSs as, in its selection process, it must choose more services, excluding some of the recognition. GDSA, although its selection process is more extensive than the other approaches, showed significant differences in its ranking process, compared to Ar_WSDS. This approach selected the vast majority of services when its QoS parameters had little variability.

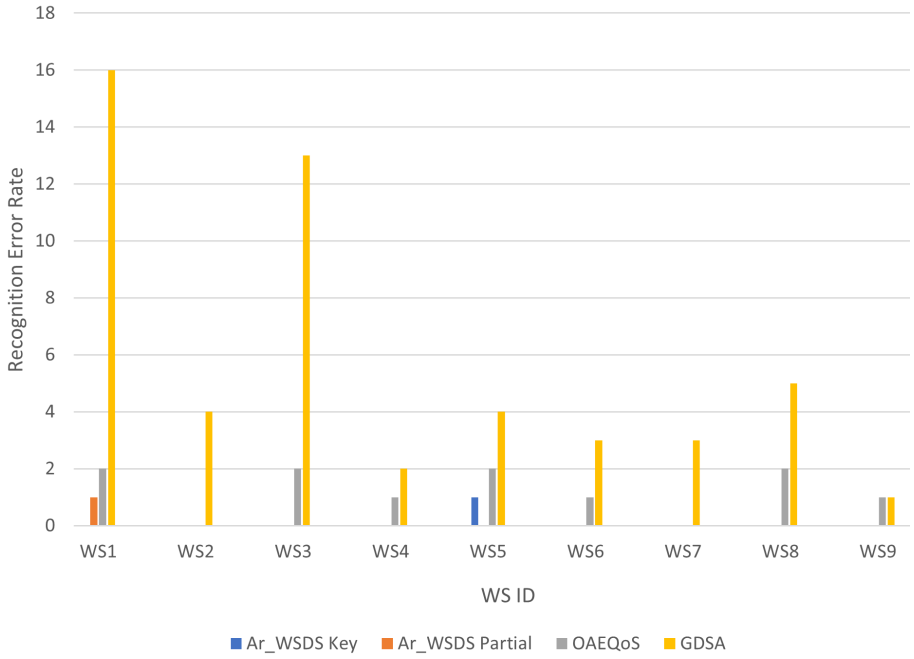

Figure 12. Error rate in the selection process according to the various algorithms.

8. Conclusions

Service discovery and selection are complex because of the availability of resources at either the design or execution stage, and their behavior can be uncertain and dynamic. Currently, research concerning service composition faces challenges underlying the basic idea of how to search and select a service in a time-efficient way. Furthermore, the dynamic environment in the cloud has led to the specification of non-functional requirements to be dynamic, and therefore a composition model can change from the design phase to its execution.

We constructed a system for WS discovery and selection based on the systemic operation of the brain considering patterns. Its process uses two strategies: key and partial signals. This recognition feature offers a method to split the problem into more straightforward jobs. Differently from the Ar2p framework, Ar_WSDS changes the condition of the key and partial signals to adjust to a WS's characteristics. Although Ar2p establishes the two recognition strategies, it is indispensable to adjust its concept in the QoS field for the case of services. Our approach can recognize multiple patterns and divide them into sub-patterns (QoS criteria) to be recognized by the strategy selected by the software developer. Our approach can recognize multiple patterns and divide them into sub-patterns to be recognized by the strategy selected by the software developer. Ar2p operates by giving priority to the key signals. Conversely, we highlight that Ar_WSDS dynamically adapts to

the recognition type for each pattern, and the system selects the services established both in the ACM and for each WS (Δ).

As a highlighted contribution, we developed Ar_WSDS based on the concept of recognizing a set of patterns that satisfy dynamic quality conditions, and the recognition strategies were updated to establish a deeper level of analysis on the QoS parameters. The dynamic recognition module creation, based on user requirements, provides an essential advantage for developers who wish to use this approach, as they can customize their searches based on the desired requirement or refine them to obtain better results.

The operating principle of recognition strategy-based Ar_WSDS offers an approach to divide the composition selection problem into functional and descriptive tasks. Researchers can quickly adapt its solution concept to other search models needing a wide variety of selection criteria.

In this work, we formalized WSs and the composition model from a mathematical perspective, allowing for the interpretation and systematization of the system in a modular architecture, thus facilitating its scalability for the addition of new QoS parameters. QoS has a dynamic nature of values due to the information supplied by the providers, so Ar_WSDS includes a module to assemble the new data without the need to alter its base programming.

According to the tests performed, KS recognition provided better performance than PS recognition, as the former limits the number of candidate WSs and, so, the search space is much smaller. However, the PS strategy was more versatile when selecting WSs, as it offers a much more comprehensive range of SPs. The recognition success rate varied with the number of constraints imposed on the model and the quality threshold placed on the composition. System performance can be affected when there are similarities in the QoS values of each WSs because the system would have to select all those services within the SP, and in this case, the KS and PS recognition could generate the same results.

According to the results, Ar_WSDS achieved excellent results in discovering and selecting WSs, compared to other existing approaches. Its strategy allows users to add more recognizers by interpreting the desired QoS parameter quickly. Overall, the proposed approach is much more effective in the selection and ranking of WSs. In particular, it allows us to analyze each QoS value independently, and as these services can be selected to achieve the WSC quality thresholds. As the system is not data-dependent, modules implemented for each recognition task are easily scalable to other systems that require element classification.

In future work, it will be essential to implementing a recognition module that evaluates the status of a service to be selected and discards those that stopped working in the providers. Likewise, it is essential to create software modules to keep the data sets updated in real-time. In order to increase the efficiency of recognition, a model could be created to create a record of the services most used by the customers; thus, the system will not need to carry out the initial search processes and will only carry them out when the services are new. Finally, a recognition module can be implemented to analyze the flow in the WSC to determine the redundancy of operations, which would allow us to refine the composition and reduce the number of services.

Author Contributions: conceptualization: M.A. and M.J.; formal analysis: M.A; methodology: M.A. and M.J.; software: M.A.; data curation: M.A; visualization: M.A. and M.J.; supervision: M.J. All authors have read and agreed to the published version of the manuscript.

Funding: The first author received a Doctoral scholarship from the Universidad Francisco de Paula Santander of Colombia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The first author expresses his deep thanks to the Universidad Francisco de Paula Santander and Universidad del Norte for their academic support.

Conflicts of Interest: The authors declare no conflict of interest

Abbreviations

The following abbreviations are used in this manuscript:

Ar_WSDS	Pattern recognition algorithm for search and selection of web services
FR	Functional requirement
NFR	Non-functional requirement
ACM	Abstract composition model
I	Input parameter
O	Output parameter
WSC	Web service composition
WS	Web service
QoS	Quality of service
SP	Solution pathway
PRTM	Pattern recognition theory of mind
WSDL	Web service description language
W3C	World Wide Web Consortium
KS	Key signal
PS	Partial signal
QWS	Quality of web service
GDSA	Generic discovery and selection algorithm

Appendix A

Figures A2 and A3 show the analysis results for the three services specified in Figure 9. The Ar_WSDS web application was used, and screenshots were captured. Each figure shows the rankings for each signal.

Global ranking	Response Time	Response Time	Throughput	Successability	Reliability	Compliance	Best Practices	Latency
BINDService - Input: Keys Recognition(4) - Partial recognition(0) Output: Web Services recognition by: key signal								
1	Rank: 1 261	Rank: 1 36	Rank: 1 0.9	Rank: 1 37	Rank: 0 60	Rank: 0 89	Rank: 0 60	Rank: 0 13.37
2	Rank: 2 266.83	Rank: 1 36	Rank: 1 0.9	Rank: 1 37	Rank: 0 60	Rank: 0 89	Rank: 0 69	Rank: 0 15.91

Figure A1. BINDService recognition example.

Global ranking	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation
DOTSGeoPhone - Input: Keys recognition (5) Output: Web services recognition by: Partial signal									
1	Rank: 0 107.4	Rank: 0 85	Rank: 0 19	Rank: 0 95	Rank: 0 73	Rank: 0 78	Rank: 0 80	Rank: 0 0.8	Rank: 0 92

Figure A2. DOTSGeoPhone recognition example.

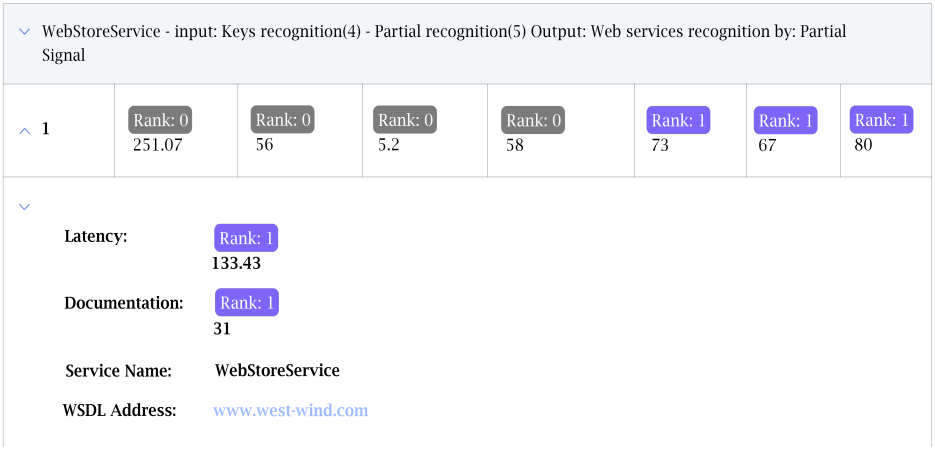

Figure A3. WebStoreService recognition example.

Appendix B

Table A1. Test results of WS ranking.

WS id	Ar_WSDS Key	Ar_WSDS Partial	OAEQoS	GDSA
1	(3,1)-(15,2)-(28,2)-(31,2)	(3,1)-(15,2)-(28,2)-(31,2)	(3,1)-(15,2)-(28,3)-(31,3)	(3,1)-(15,12)-(28,6)-(31,4)
2	(5,1)	(5,1)	(5,1)	(5,3)
3	(19,1)-(2,1)-(8,1)-(20,2)-(10,2)-(9,2)	(19,1)-(2,1)-(8,1)-(20,2)-(10,2)-(9,2)	(19,2)-(2,3)-(8,0)-(20,2)-(10,2)-(9,0) *	(19,1)-(2,2)-(8,3)-(20,2)-(10,1)-(9,2)
4	(2,1)	(2,1)	(2,1)	(2,1)
5	(8,1)-(2,1)-(11,2)-(4,2)	(8,1)-(2,1)-(11,2)-(4,3)	(8,1)-(2,0)-(11,2)-(4,0) *	(8,1)-(2,1)-(11,2) -(4,2)
6	(1,1)-(5,2)	(1,1)-(5,2)	(1,1)-(5,0) *	(1,1)-(5,2)
7	(2,1)-(3,1)-(11,2)-(7,3)-(6,3)	(2,1)-(3,1)-(11,2)-(7,3)-(6,2)	(2,1)-(3,1)-(11,2)-(7,3)-(6,2)	(2,1)-(3,2)-(11,3)-(7,3)-(6,2)
8	(10,1)-(4,1)-(11,2)	(10,1)-(4,1)-(11,1)	(10,0)-(4,1)-(11,0) *	(10,1)-(4,2)-(11,2)
9	(3,1)	(3,1)	(3,1)	(3,1)

* Solution pathways along those in which one or more services were missing.

References

1. Hayyolalam, V.; Kazem, A.A.P. A systematic literature review on QoS-aware service composition and selection in cloud environment. *J. Netw. Comput. Appl.* **2018**, *110*, 52–74. [CrossRef]
2. Dahan, F.; El Hindi, K.; Ghoneim, A.; Alsalman, H. An Enhanced Ant Colony Optimization Based Algorithm to Solve QoS-Aware Web Service Composition. *IEEE Access* **2021**, *9*, 34098–34111. [CrossRef]
3. Wang, H.; Wang, X.; Hu, X.; Zhang, X.; Gu, M. A multi-agent reinforcement learning approach to dynamic service composition. *Inf. Sci.* **2016**, *363*, 96–119. [CrossRef]
4. Qi, J.; Xu, B.; Xue, Y.; Wang, K.; Sun, Y. Knowledge based differential evolution for cloud computing service composition. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 565–574. [CrossRef]
5. Nacer, H.; Beghdad Bey, K.; Djebari, N. Migration from web services to cloud services. *Lect. Notes Comput. Sci.* **2017**, *10542*, 179–192._16. [CrossRef]
6. Ramirez, A.; Parejo, J.A.; Romero, J.R.; Segura, S.; Ruiz-Cortés, A. Evolutionary composition of QoS-aware web services: A many-objective perspective. *Expert Syst. Appl.* **2017**, *72*, 357–370. [CrossRef]
7. Mishra, T.; Raj, G. QoS implementation in Web Services selection and ranking using data analysis. In Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science Engineering Confluence, Noida, India, 12–13 June 2017; pp. 537–542. [CrossRef]
8. Puerto Cuadros, E.G.; Aguilar Castro, J.L. Un algoritmo recursivo de reconocimiento de patrones. *Rev. Técnica Fac. Ing. Univ. Zulia* **2017**, *40*, 95–104.
9. Ghobaei-Arani, M.; Rahmanian, A.A.; Aslanpour, M.S.; Dashti, S.E. CSA-WSC: Cuckoo search algorithm for web service composition in cloud environments. *Soft Comput.* **2018**, *22*, 8353–8378. [CrossRef]
10. Di Martino, B.; Cretella, G.; Esposito, A. Cloud services composition through cloud patterns: A semantic-based approach. *Soft Comput.* **2017**, *21*, 4557–4570. [CrossRef]

11. Rangarajan, S. Qos-Based Web Service Discovery And Selection Using Machine Learning. *EAI Endorsed Trans. Scalable Inf. Syst.* **2018**, *5*. [CrossRef]
12. Chakravarthy, D.G.; Kannimuthu, S. Extreme Gradient Boost Classification Based Interesting User Patterns Discovery for Web Service Composition. *Mob. Netw. Appl.* **2019**, *24*, 1883–1895. [CrossRef]
13. Sha, J.; Du, Y.; Qi, L. A user requirement oriented web service discovery approach based on logic and threshold petri net. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1528–1542. [CrossRef]
14. Wu, Y.; Yan, C.; Ding, Z.; Liu, G.; Wang, P.; Jiang, C.; Zhou, M.C. A Multilevel Index Model to Expedite Web Service Discovery and Composition in Large-Scale Service Repositories. *IEEE Trans. Serv. Comput.* **2016**, *9*, 330–342. [CrossRef]
15. Hasnain, M.; Pasha, M.F.; Ghani, I.; Mehboob, B.; Imran, M.; Ali, A. Benchmark dataset selection of Web services technologies: A factor analysis. *IEEE Access* **2020**, *8*, 53649–53665. [CrossRef]
16. Rathore, M.; Suman, U. Evaluating QoS parameters for ranking Web service. In Proceedings of the 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, India, 22–23 February 2013; pp. 1437–1442. [CrossRef]
17. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]
18. Khalid, S.; Khalil, T.; Nasreen, S. A Survey Of Feature Selection And Feature Extraction Techniques In Machine Learning. In Proceedings of the 2014 Science and Information Conference, London, UK, 27–29 August 2014; pp. 372–378. [CrossRef]
19. Oliveri, P.; Malegori, C.; Mustorgi, E.; Casale, M. Qualitative pattern recognition in chemistry: Theoretical background and practical guidelines. *Microchem. J.* **2021**, *162*, 105725. [CrossRef]
20. Escobar, C.A.; Morales-Menendez, R. Machine learning and pattern recognition techniques for information extraction to improve production control and design decisions. In Proceedings of the 2017 Industrial Conference on Data Mining (ICDM), New York, NY, USA, 12–13 July 2017; pp. 286–300._23 [CrossRef]
21. Paolanti, M.; Frontoni, E. Multidisciplinary Pattern Recognition applications: A review. *Comput. Sci. Rev.* **2020**, *37*, 100276. [CrossRef]
22. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Umar, A.M.; Linus, O.U.; Arshad, H.; Kazaure, A.A.; Gana, U.; Kiru, M.U. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access* **2019**, *7*, 158820–158846. [CrossRef]
23. Chen, C.H. *Handbook of Pattern Recognition and Computer Vision*, 5th ed.; World Scientific: Dartmouth, MA, USA, 2016. [CrossRef]
24. Puerto, E.; Aguilar, J.; Reyes, J.; Sarkar, D. Deep learning architecture for the recursive patterns recognition model. *J. Phys.* **2018**, *1126*, 12035. [CrossRef]
25. Puerto, E.; Aguilar, J.; Perez, B. Análisis de la teoría de la mente humana basada en el reconocimiento de patrones. In Proceedings of the 2014 Congreso Internacional en Innovación y Apropiación de las Tecnologías de la Información y las Comunicaciones(CIINATIC), Bucaramanga, Colombia, 28–30 October 2014; pp. 101–106.
26. Viriyasitavat, W.; Bi, Z. Service selection and workflow composition in modern business processes. *J. Ind. Inf. Integr.* **2020**, *17*, 100126. [CrossRef]
27. Wang, S.; Zheng, Z.; Sun, Q.; Zou, H.; Yang, F. Cloud model for service selection. In Proceedings of the Computer Communications Workshops (INFOCOM WKSHPS), Shanghai, China, 10–15 April 2011; pp. 666–671.
28. Al-Masri, E.; Mahmoud, Q.H. Investigating web services on the world wide web. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; pp. 795–804.
29. Al-Masri, E. A Quality-Driven Approach for Ranking Web Services. In *New Trends in Networking, Computing, E-learning, Systems Sciences, and Engineering*; Khaled, E., Sobh, T., Eds.; Springer International Publishing: Cham, Swizerlands, 2015; pp. 599–606.
30. Devi, M.M. Survey on Choreography for Web Services. *Int. J. Future Revolut. Comput. Sci. Commun. Eng.* **2018**, *4*, 149–155.
31. Baryannis, G.; Kritikos, K.; Plexousakis, D. A specification-based QoS-aware design framework for service-based applications. *Service Oriented Comput. Appl.* **2017**, *11*, 301–314. [CrossRef]