From Business Process to Cloud Application

Hamdi GABSI, RIADI Laboratory. National School of Computer Sciences, University of Manouba, la Manouba, Tunisia, hamdi.gabsi@ensi-uma.tn
Rim DRIRA, RIADI Laboratory. National School of Computer Sciences, University of Manouba, la Manouba, Tunisia, rim.drira@ensi-uma.tn
Henda HAJJAMI BEN GHEZALA, RIADI Laboratory. National School of Computer Sciences, University of Manouba, la Manouba, Tunisia, henda.benghezala@ensi-uma.tn
Stéphane DUCASSE, Inria Lille-Nord Europe, F- 59000 Lille, France, stephane.ducasse@inria.fr

Abstract

Business Process (BP) development can be defined as the process of constructing a workflow application by composing a set of services performing BP's activities. In this respect, Cloud Services (CSs) are being increasingly used in BP development to ensure a high level of performance with a low operating cost. Although large companies may benefit from CSs' advantages, Small and Medium-sized Enterprises (SMEs) and startups are falling behind in cloud usage due to missing Information Technology competence, (IT-competence). The crucial challenge facing SMEs and startups in cloud-based BP development is to effectively address the so-called business and IT alignment issue. It represents the alignment between two different domains; one that entails technical cloud resource requirements and another comprising business-level. Formerly, we present this issue as a discovery challenge of suitable CSs performing abstract BP's activities. To address this challenge, firstly, we introduce the concept of cloud-aware BP by proposing a Domain-Specific Language (DSL) named "BP4Cloud" to enrich BP modeling and cover both business and technical requirements. Secondly, we propose an Activity-Services Matching algorithm that automates the discovery of CSs performing BP's activities.
As a part of the evaluation, we set up by clarifying the specification of BP4Cloud elements through a proof of concept implementation applied on a real BP. Then, we proceed by evaluating the precision and recall of our Activity-Service Matching algorithm.

Keywords

Cloud services discovery, Business process development, Activity-Services Matching.

Introduction

A business process is the combination of a set of activities which are performed in coordination within an organizational and technical environment. These activities jointly realize a business goal [Carrillo and Sobrevilla, 2017]. The BP development leads to define a workflow application which consists of coordinated executions of multiple activities that require access to high performance IT services [Carrillo and Sobrevilla, 2017]. For this reason, companies are interested in cloud computing environments which provide on-demand high performance IT services based on a pay-as-you-go model, allowing them to reduce the management costs, improve their productivity and decrease the time to market.

Despite these advantages, SMEs and startups are falling behind in cloud usage due to missing IT competences. In fact, BP designers and technical developers usually have to cope with alignment challenges between business aspects and CSs needed for BP development. This challenge is presented through two perspectives, on the one side, we introduce the concept of cloud-aware BP which presents an enrichment of BP by supporting cloud requirements modeling. On the other side, we handle the mapping of BP's activities to concrete CSs required to perform the BP.

Despite the existence of two widely adopted standards for process models and execution like Business Process Model and Notation "BPMN" [Geigera et al., 2018] and Web Services Business Process Execution Language "WS-BPEL" [Sun et al., 2019], there is no general agreement on the mapping and alignment between the business modeling side and the performing of abstract business's activities side. Several efforts have investigated the business and IT alignment challenge in cloud environments such as [Nacer et al., 2019], [Martino et al., 2018] and [Nagarajan et al., 2018]. However, fundamental issues remain for further investigations, mainly, a discovery process ensuring an automatic mapping from business activities to CSs.

The contributions of this paper can be summarized as follows:

- Introducing the concept of cloud-aware BP by proposing a DSL named ”BP4Cloud”, an extension realized in full compliance with BPMN meta-model to cover both business and technical requirements and pave the way to the CSs discovery step.
- Proposing an Activity-Services Matching algorithm to automate the mapping between business activities and CSs.
- Demonstrating the effectiveness of the proposed algorithm through experimental validation.

The remainder of this paper is organized as follows: Section 2 presents a motivating scenario. Section 3 details BP4Cloud language. Section 4 presents the Activity-Services Matching algorithm. Section 5 illustrates the validation of our contributions. In Section 6, we discuss the related work. Section 7 provides concluding remarks and outlines our ongoing works.

Motivating Example

Our main focus in this paper is to assist the modeling and discovery of cloud resources required for BPs development. To effectively achieve our purpose, we need to outline a substantial interaction between business designers and technical developers during the resources modeling and resources discovery.

The resources modeling presents the specification of abstract entities that carry out the work related to activities which are in our context CSs [Carrillo and Sobrevilla, 2017].

The resources discovery presents the identification of concrete CSs needed to (i) the deployment and (ii) the performing of abstract BP’s activities. The deployment of the BP requires identifying suitable infrastructure and platform services. These services provide virtualized resources and software tools needed for application development. The performing of BP activities requires a mapping between abstract business activities and concrete software services.

To illustrate, let us suppose the following scenario presented in Figure 1. This scenario describes a simple BPMN process for online bank accounts opening: which initiates with an application request sent from a customer. The first activity consists on checking the customer summary, if a similar request is already in process, his/her request will be rejected. Otherwise, the process evaluates the customer application, if the application is approved, a provisional account Id is generated and sent. In order to develop the bank account opening process using CS, two major steps should be conducted:

First, we need to define technical cloud resources, essentially, IaaS and PaaS requirements. For instance, the evaluation application activity requires a computing resource to evaluate the customer’s application. The check customer summary activity requires a database server to verify if the customer has already an account Id. The whole process may require a load-balancer to distribute the application traffic across the allocated compute resources. However, BPMN, which is the most used standard for the high-level description of BPs, supports neither the modeling of the required cloud resources nor their configuration features. This fact presents an important challenge for business designers.

The second step consists of the discovery of software services performing abstract activities. For instance, the sending account Id activity can be performed by a service which sends notifications or emails. Our main purpose is to assist business designers in the aforementioned steps.

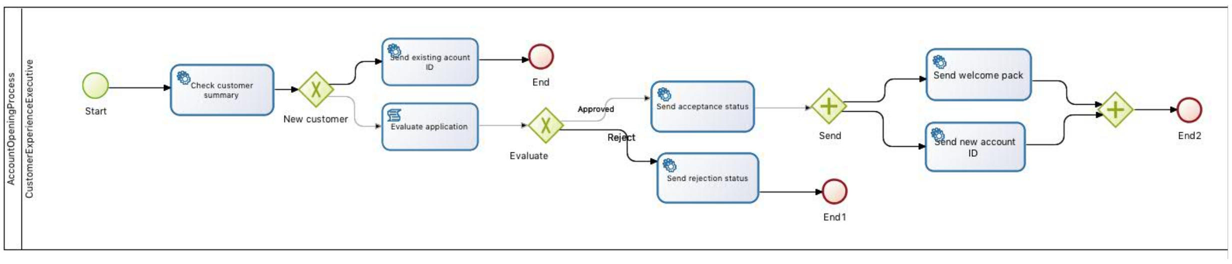


Fig 1. BPMN Process Example.

BP4Cloud Language

BP4Cloud consists of a set of extension elements that allow the attachment of cloud resource perspectives to the BPMN standard. By enriching BPMN with CSs (Cloud Services) covering both business and technical requirements, we define a cloud-aware BP. BP4Cloud considers two aspects:

- First, technical resource requirements which are IaaS and PaaS services allowing to define the run-time environment.
- Second, business specifications which assist SaaS services discovery to perform abstract business activities.

Concretely, we specified these proposed extension definitions in an xsd document derived from the BP4Cloud meta-model. By importing our xsd extension in the BPMN document, we extend the BPMN document by putting into the core of its tag the corresponding extension elements.

Technical Cloud Resources Requirements

The first aspect considered by BP4Cloud is the modeling of IaaS and PaaS requirements. IaaS requirements cover three types of resources which are: computing, storage, and network resources.

- The computing aspect provides resources able to deploy and run service activities or execute script activities (e.g., virtual machines) [Nacer et al., 2019].
- The storage meta-class presents the requirement of information recording resources in data storage devices. There are different ways to use the storage resource depending on activities requirements. Namely, the online storage state manages the shared data. The backup state manages private data.
- The network meta-class presents network resources providing mechanisms that are used for communication [Nacer et al., 2019]. The network type denotes an interconnection resource (e.g., a virtual switch).

The PaaS requirements cover the virtual appliances' requirements that present the middleware components needed for the execution of activities such as web server, application server, etc. Figure 2 illustrates the meta-model presenting the infrastructure requirements.
Our goal is to assist non-cloud experts in identifying their technical requirements. Thus, we specified main cloud technical requirements. Such requirements can be exhaustively extended.

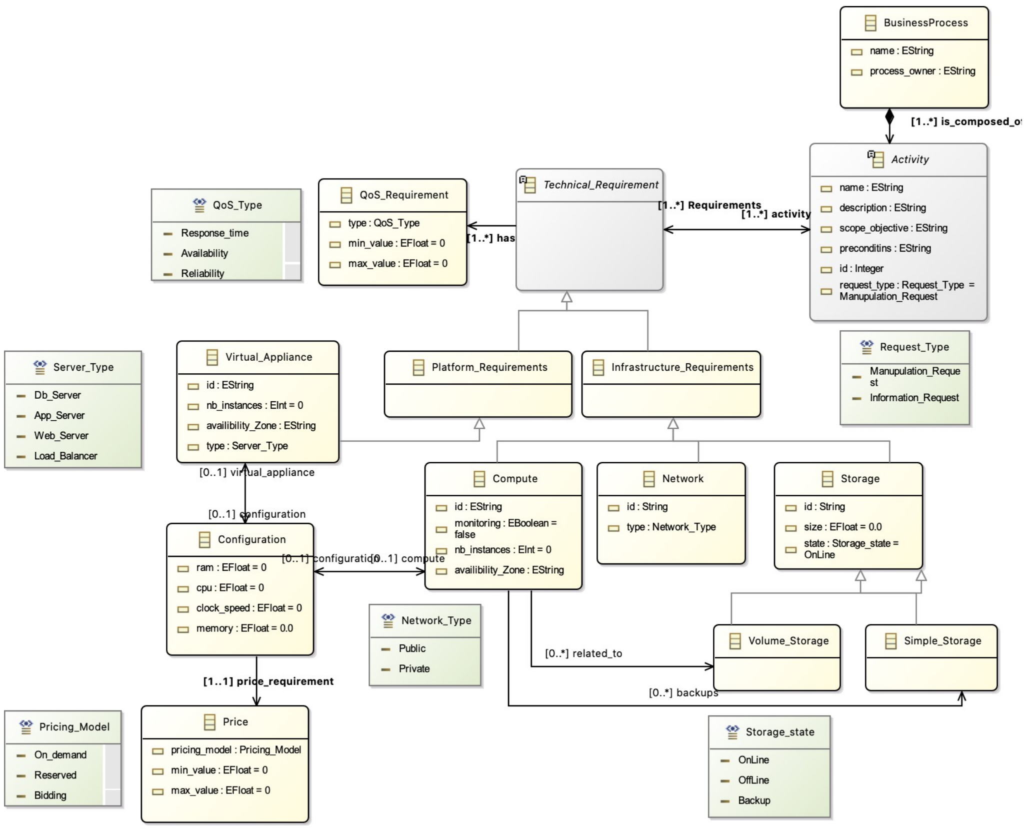


Fig 2. BP4Cloud Meta-Model.

Software Cloud Resources Requirements

The second aspect considered by BP4Cloud is the required software/application services. In practice, these services are identified using our proposed Activity-Services Matching algorithm. To improve the performance and the results of the algorithm, we need accurate specifications related to the business logic of BP's activities. For these reasons, we introduce the Business Logic meta-class which classifies activities according to their business logic. This classification is basically inspired by Workflow Pattern Activity (WPA) [Workflow Resource Patterns, 2018]. WAP describes a business function frequently found in BPs [Martino et al., 2017]. In fact, BP can be seen as several connected process fragments each one presents a particular business function. Thom et al. [Lucineia et al., 2009] performed a manual analysis to identify relevant WAPs as well as their cooccurrences within a collection of 214 real wold BPs. Based on the frequency of appearance and potential reuse of a business function in the analyzed models, the authors define seven WPAs which are : Approval, Question-answer, Uni/Bi-directional Performative, Information Request, Notification, and Decision Making.

In our proposal, a CS classification is conducted based on WPAs to identify candidate services for a particular business logic. This fact can improve the precision of the algorithm by identifying service classes that share the same business logic as the activity. To do so, we propose a modified K-means algorithm based on CS descriptions. To have meaningful classification adapted to our context, first, we left frequent and rare words extracted from services description unclassified to avoid overfitting.
Second, we enhance the cohesion and correlation conditions defined in the standard K-means algorithm by quantifying the cohesion and correlation of classes based on a semantic functional similarity (which is presented in the next section) instead of Euclidean distances used in the standard K-means algorithm.
Third, to ensure that we obtain classes with high cohesion, we only add an item (in our case a service) to a class if it satisfies a stricter condition, called cohesion condition. Given a class C, an item 'i' is called a kernel item if it is closely similar to at least half of the remaining items in C. Our cohesion condition requires that all the items in the class be kernel items. Formally;

$$i \in C \Longrightarrow (\forall j \in C, i \neq j), Sim(i,j) \geq \frac{\|C\| - 1}{2}$$

We illustrate the major steps of the modified K-means algorithm as follows: (what is srepeat?) I cannot read the identify of function.

```
Modified K-means Algorithm:
Input: classes scopes set (S = s1, s2, ..., sn ) k the number of classes
Output: k classes
Let sim be the similarity function
C = {c1, c2,... , ck } (set of classes centroids)
L = {L(si)|i = 1, 2, ..., n}(set of classes labels)
for all ci in C do
        ci <-- sj {Initialize Centroid (ci) }
        end for
for all si in S do
        l(si) <-- index_max_Sim(si,cj)
        end for
Centroid_Change <--False, cohesion_condition() <--True
srepeat
        for all ci in C do UpdateCentroid(ci)
        end for
for all si in S do
        M <-- index_max_Sim(si,cj)
        if M /= l(si) then
                l(si) <-- M
                Centroid_Change <-- True
        end if
        Verify(cohesion_condition(si, sl))
        l in (1, 2, ..., n) l /= i
end for
until (Centroid_Change == False and Verify(cohesion_condition()))
```

Figure 3 illustrates activities' business logics.

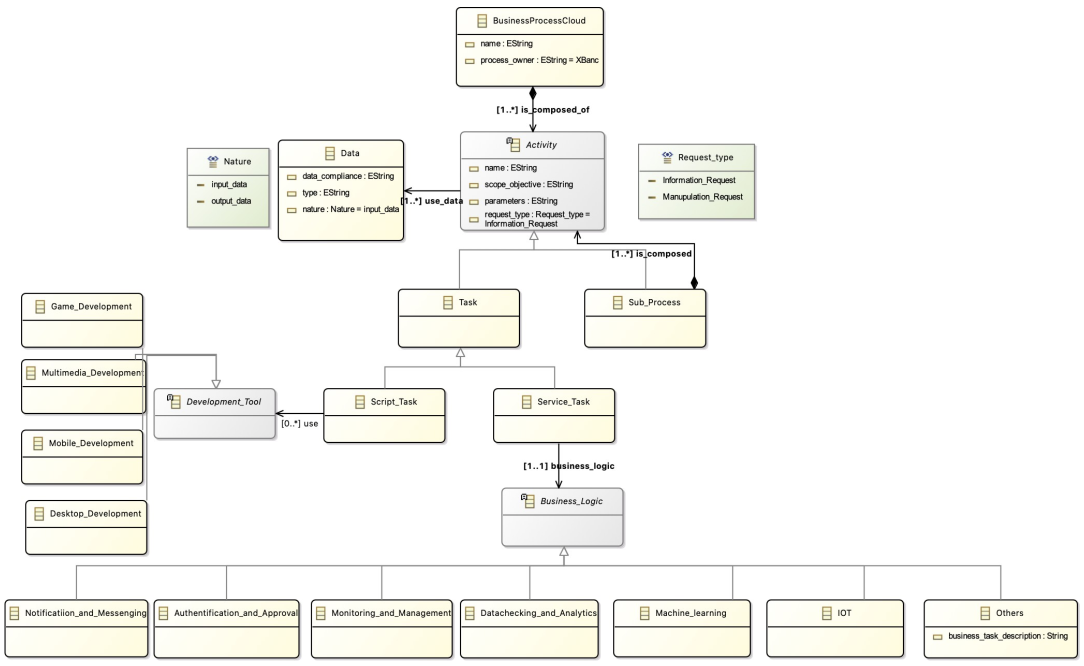


Fig 3. BP4Cloud Meta-Model

Activity-Services Matching Algorithm

After modeling the required cloud resources, we need to particularly discover concrete services that will be invoked to perform BP's activities. To do so, we propose the Activity-Services Matching algorithm which is accomplished through two main steps: the technical matching and the business matching. Figure 4 illustrates an overview of the matching algorithm steps.

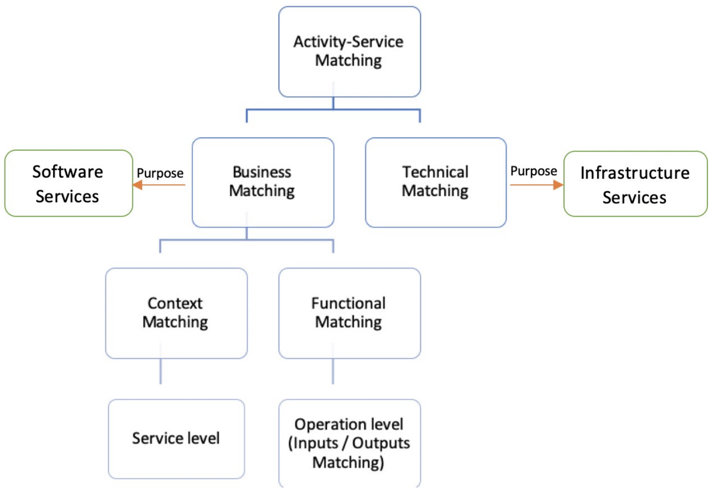


Fig 4. Activity-Services Matching steps

Our algorithm uses real services referenced in a cloud registry. We proposed a consistent registry, named ULIT (Unified cLoud servIces regisTry), where services offered by different cloud providers, such as Amazon, IBM, and Google, are collected, unified and classified based on their functional features. ULIT was reviewed and accepted by Elsevier Mendeley Data [Elsevier Mendeley Data , 2019]. It is publicly available on [ULIT , 2019].

Technical Matching

The main purpose of the technical matching is discovering the infrastructure environment based on technical requirements specified by BP4Cloud. To do so, we establish, first, a mapping relationship between ”agnostic cloud resources” specified in BP4Cloud and ”concrete cloud resources” referenced in ULIT. Figure 5 illustrates this mapping.

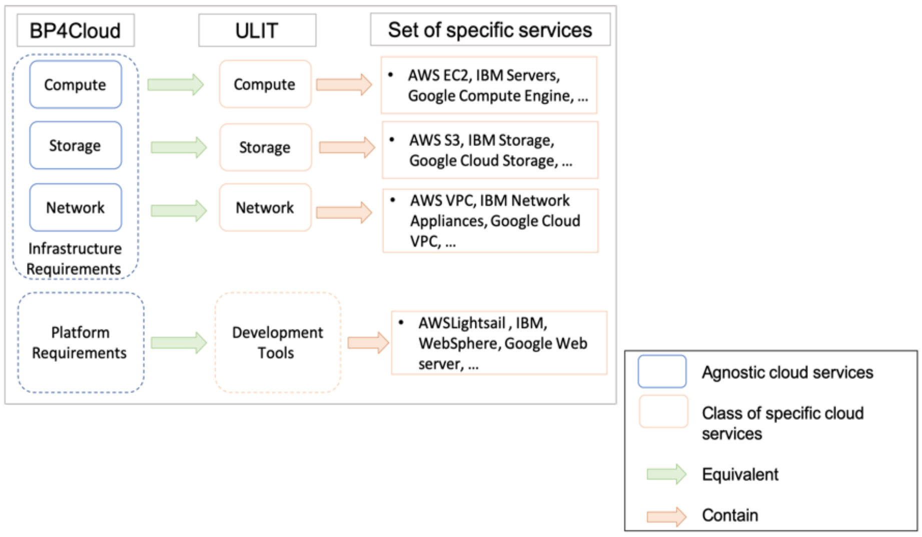

BP4Cloud
ULIT
Set of specific services
Compute
Storage
Network
Infrastructure Requirements
Compute
Storage
Network
• AWS EC2, IBM Servers, Google Compute Engine, ...
• AWS S3, IBM Storage, Google Cloud Storage, ...
• AWS VPC, IBM Network Appliances, Google Cloud VPC, ...
Platform Requirements
Development Tools
• AWSLightsail , IBM, WebSphere, Google Web server, ...
Agnostic cloud services
Class of specific cloud services
Equivalent
Contain

Fig 5. Technical Matching

However, this is not enough to precisely define the infrastructure environment. Indeed, IaaS services need to be correctly configured. Several and conflicting criteria have to be considered such as VCPU, RAM, etc. No single service exceeds all other services in all criteria but each service may be better in terms of some of the criteria. Therefore, we consider IaaS services discovery and configuration as a MultiCriteria Decision Making (MCDM) problem. In our previous works [Gabsi et al., 2018] and [Gabsi et al., 2019], we have clearly addressed the technical matching.

Business Matching

The business matching is conducted through two steps. The context matching and the functional matching. Figure 6 illustrates the business matching steps.

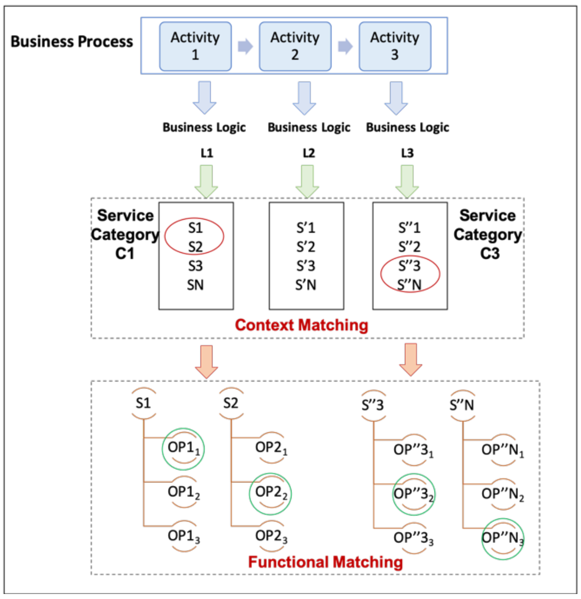


Fig 6. Business Matching.

Context Matching

The context matching discovers CSs sharing the same business context as BP's activity. To identify the business

context of BP's activities, we annotate each activity by scope and objective that clearly define the business context such as Send Notification, Database Access. The scope and objective are defined by the business designer.
Defining the business context of a CS is more challenging due to its heterogeneous description and the abundant use of adjectives for commercial purposes. To deal with this challenge, we extract relevant keywords presenting the functional features of CSs from its provided descriptions in supplier's web portals. We use a natural language processing tool named Stanford Parser [Marneffe and Manning, 2015]. The Stanford Parser can identify the grammatical structure of each sentence of service descriptions by creating grammatical relations or type dependencies among elements in the sentence. These dependencies are called Stanford Dependencies SDs [Marneffe and Manning, 2015]. In our work, we model keywords as a set of binary relations < action, object >, where the action is a keyword presenting a verb, which denotes the functional feature of the service. The object is a keyword presenting a noun, which denotes the entities affected by the action. Then, we use the SD sets to properly identify the grammatical relations between < action, object >. In fact, each SD is a binary relation between a governor (also known as a regent or a head) and a dependent [Marneffe and Manning, 2015].

After defining services functional features, the context matching is based on the semantic similarity between service's keywords and activity's scope & objective. The semantic similarity *SIM(S₁, A₁)* is inspired by [Lin, 1998a]. Indeed, the authors prove the relevance of the proposed similarity formula in the context of word pair similarity. The main asset of this work is defining similarity in information theoretic terms which ensure the universality of the similarity measure.

We denote $S_1 = \{s_{1,1}, s_{1,2}, s_{1,3},..., s_{1,n}\}$ *a* service business features, where s1,i is the pair < action, object > and $|S_1|$ the cardinality of S_1, (the number of pairs $s_{1,i}$). We denote $A_1 = \{a_{1,1}, a_{1,2}, a_{1,3}, ..., a_{1,n}\}$ *the* activity's scope and objective. The similarity *SIM(S₁, A₁)* is the sum of the similarities between each pair s_{1i} of S_1, and the pairs $\{a_{1_1}, a_{1_2}, a_{1_3},...,a_{1_in}\}$ of A_1, normalized by the cardinality of S_1. Formally, we obtain:

$$SIM(S_1, A_1) = \frac{\sum_{i=1}^{|S_1|}\left(\frac{\sum_{j=1}^{|A_1|} SIM\ (s_{1i}, a_{1j})}{|A_1|}\right)}{|S_1|}$$

where $S(s_{1i}, a_{1j})$ is the pair similarity. We define the pair similarity as follows:

- Let A_{s1} and A_{c1} respectively denote the actions in the pair s_1 and a_1;
- OS_{i1} and OA_{i1} respectively denote objects in the pair s_1 and a_1;
- ω_1, ω_2 denote the weight of the action part and the object part. We suppose that some predefined action has a higher weight such as; offer, provide, deliver, etc.;
- m is the minimum number of objects of the pair s_1 and a_1 (min number objects(s_1,a_1)) whereas n is the maximum number of objects of the pair a_1 and s_1 (max number objects(a_1 ,s_1)).

The pair similarity is calculated as :

$$SIM(a_1, s_1) = \omega_1 SIM(A_{S1}, A_{c1}) + \ \omega_2 \frac{\sum_{i=1}^{m} SIM(OS_{i1}, OA_{i1})}{n}$$

where $SIM(OS_{i1}, OA_{i1})$ is words similarity. We use Jacard Similarity Coefficient [Niwattanakul et al., 2013] to calculate the word similarity. In fact, the Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the union divided by the size of the intersection of the sample sets.

$$J(E, F) = \frac{E \cup F}{E \cap F}$$

where E and F are two given sets of words. We create a feature set F(w) for each word 'w' (i.e., for each object which is presented by a word in our context) containing the synonym set, generic word and interpretation of the word w. F(w) is created using BabelNet. Based on [Lin, 1998b], we define the similarity between the two words as follows:

$$SIM(\omega_1, \omega_2) = \frac{2 \times I(F(\omega_1) \cap F(W_2))}{I(F(\omega_1) + F(W_2))}$$

where I(S) represents the amount of information contained in a set of features S. I(S) is calculated as:

$$I(S) = -\sum_{f \in S} log P(f)$$

The probability $P(f)$ can be estimated by the percentage of words that have the feature f among the set of words that have the same part of speech in the entire BabelNet library database. If the semantic similarity $SIM(S_1, A_1)$ is greater than a specific threshold (experimentally fixed in our work as 0.5), S_i is considered as candidate for the functional matching.

Functional Matching

The output of the context matching is a set of candidate CSs that share the same business context as the activity. The functional matching aims to identify the CS's operations that can perform the BP's activity. Inspired by [TRAN et al., 2009], we perform the functional matching by evaluating how a service operation fulfills an activity requirement. This is performed by answering two symmetric questions: (1) how the activity can fulfill required inputs for the service operation; and (2) how the service operation can fulfill expected outputs of the activity. To do so, first, we classify CSs operations and business activities into two categories: information request (IR) and manipulation request (MR). Information requests (IR) aim to retrieve diverse information regarding a particular request such as get attributes(), list methods() etc. Manipulation requests (MR) present several modification functions such as: create(), recommend(), etc. Based on this classification, we identify candidate operations according to the activity request type. We define a matched operation if two main elements are verified: a matched context and matched inputs/outputs.

A matched context is defined through two steps. First, we take advantage of the structure of CSs operations' naming, which is typically action object, to calculate the semantic similarity between the operation and the activity actions. If this similarity is greater than a fixed threshold, the second step consists on verifying if the operation object is a generic word of the activity object, that means the activity object satisfies the relationship *"is a"* an operation object. We consider an operation is a candidate for the inputs/outputs matching if it satisfies the two above-mentioned conditions.
The input/output matching consists of a matching level and matching degree. The matching degree consists of four elements: SIF: service operation input fulfillment, AIR: activity input redundancy, AOF: activity output fulfillment, and SOR: service operation output redundancy. SIF and AOF are, respectively, the ratio of the fulfillment of the service operation inputs and activity outputs by the activity inputs and service operation outputs. AIR and SOR are, respectively, the ratio of redundancy (unused) of the activity inputs and service operation outputs in fulfilling the service operation inputs and activity outputs. SIF, AIR, AOF, and SOR have values in [0.1]. Formally:

$$SIF = \frac{Matched\_Input\_Count}{Operation\_Input\_Size}, \qquad SOR = \frac{Unused\_Operation\_Otput}{Operation\_Otput\_Size}$$

$$AIR = \frac{Unused\_Activity\_Input}{Activity\_Input\_Size}, \qquad AOF = \frac{Matched\_Output\_Count}{Activity\_Output\_Size}$$

A matching level can be precise, over, partial, or mismatch and it is specified based on SIF, AIR, AOF, and SOR in the following rules:

- Precise: if (SIF = 1 ∧ AOF = 1) ∧ (AIR = 0 ∧ SOR=0)
- Over if (SIF=1 ∧ AOF=1) ∧ (AIR>0 ∨ SOR> 0)
- Partial if (SIF >= iMT ∧ SIF < 1) ∧ (AOF >= oMT ∧ AOF < 1)
- Mismatch if SIF < iMT ∨ AOF < oMT

iMT and oMT are customized matching thresholds. Depending on a particular business context, we can set suitable values for iMT and oMT to get more or less partial matched services, e.g., they can be set to 0.5. If there are more than one service operations which are matched with the activity then the list of matched service operations is sorted according to the following rules:

- Precise > Over > Partial > Mismatch
- If two operations are both over match then
  - The smaller the value SOR is the better the matched operation is.

    - If they have the same value SOR then the smaller the value AIR is the better the matched operation is.
- If two operations are both partial match then
    - The larger the value AOF is the better the matched operation is.
    - If they have the same value AOF then the larger the value SIF is the better the matched operation is.
    - If they have the same values AOF and SIF then we apply the rules of the values SOR and AIR as in case of over match above.
- If two operations are both mismatch then they are considered the same.

The following algorithm illustrates the Activity- Services Matching process.

```
Activity-Services Matching Algorithm (Please indent)
Let S be a Cloud Service.
Resources.
{Context Matching}
Select S where S.Class == A.BusinessLogic
If (Sim (A.Scope_Objective, S.Keywords_Set)>=0.5) then
        Add(Candidate_Services,S)
{Functional Matching}
For each S in Candidate_Services do
        Select S.Operations_Set where A.Request_Type == S.Operation_Category
        If Sim(A. Verb, S.Operation_Verb) >= 0.7 then
                If (A.Object is_a (S.Operation_Object) Add(Candidate_Operaton , S.Operation)
                Endif
        Endif
Endfor
matchCount=0
unusedCount = 0
For each op in Candidate_Opererations do
        For each inp in op.Input_set do
                m = match (inp, A.input) (the match function checks the compatibility between
                data types.)
                if m != "mismatch" then
                        matchCount ++
                Endif
        Endfor
Endfor
For each a_input in A.input do
        If a_input.used == "false" then
                unusedCount ++
        Endif
Endfor
(Service Input Fulfillment)
Sif = matchCount /S.Input.size();
(Activity Input Redundance)
Air = unusedCount /A.Input.size();
If Sif==1 and Air==0 then
        matchingInput.Level = "precise"; elseifSif==1andAir>0then matchingInput.Level = "over";
elseif  Sif < 1 and Sif >= iMT(0,5) then
matchingInput.Level = "partial";
elseif Sif < iMT(0,5) then
        matchingInput.Level = "mismatch";
Endif
(We apply the same algorithm for matching
the outputs as matching the inputs.)
If (matchingInput.Level == "precise" and matchingOuput.Level == "precise") then
        matchingOperation.Level == "precise";
```

```
elseif (matchingInput.Level == "precise" and matchingOutput.Level == "over") or
(matchingInput.Level == "over" and matchingOutput.Level == "precise") or (matchingInput.Level ==
"over" and matchingOutput.Level == "over") then
        matchingOperation.Level = "over";
elseif (matchingInput.Level == "partial" and matchingOutput.Level != "mismatch") or
(matchingInput.Level != "mismatch" and matchingOutput.Level == "partial") then
        matchingOperation.Level = "partial";
elseif (matchingInput.Level == "mismatch" or matchingOutput.Level == "mismatch") then
        matchingOperation.Level = "mismatch";
Endif
```

Application and Performance Analysis

To illustrate our approach, we clarify the specification of BP4Cloud elements applied on the use case, ”Online Bank Accounts Opening”. Then, we proceed evaluating the overall performance of our Activity-Service Matching Algorithm.

BP4Cloud Evaluation

We present an extract of BP4Cloud file specifying the activity ”Check customer summary” requirements. The activity requires a compute server, as IaaS requirement, and a database server as PaaS requirements. Its business logic is ”Datachecking and Analytics.

```
<is_composed_of
     xsi:type="CloudBusinessProcess: Service_Task"
     name="Check customer summary"
     description="Verify if a customer has already made an application earlier"
     key_words="Check summary", "Verify application", "Validate Summary"
     preconditins=""
     effects=""
     id="1">
   <use_data
       href="Data.xmi#/" input_data_type="
            String" output_data_type="Boolean "/>
   <requirements
       xsi:type="CloudBusinessProcess: Data_Base_Server"
       href="Data_Base_Server.xmi#/">
   <requirements
       xsi:type="CloudBusinessProcess:Compute"
       href="Compute.xmi#/" id="1" vmsize="
            Meduim">
   <business_logic
       href="Datachecking_and_Analytics.xmi #/"/>
```

To practically evaluate our specification, we measure the complexity of two process models, one using the standard BPMN and the other using BPMN extended by BP4Cloud. The complexity of a process model can be defined as the degree to which a BP is difficult to analyze and understand. [Huber et al., 2018]. Huber et al. [Huber et al., 2018, propose to categorize BP complexity metrics. We consider the following metrics:

- The number of activities and control flow elements in a process metric (NOAC). It counts the activities and control flow elements of a process, which are the gateways in BPMN.
- McCabe’s Cyclomatic Complexity metric (MCC) measures the number of control paths through the process. MCC is defined to be $e - n + 2$, where e is the number of edges and n is the number of nodes in the control flow graph.
- Control flow Complexity metric (CFC) is defined as the number of mental states that have to be considered when a designer develops a process. It is calculated as $C_{XOR} + C_{OR} + C_{AND}$, where C_{XOR} is complexity of XOR-split (equals to number of branches that can be taken), C_{OR} complexity of OR-split (equals to number of states that may arise from the execution of the split), and C_{AND} complexity of

AND-split (always equals 1). The higher the value of C_{XOR}, C_{OR}, and C_{AND}, the more complex is the process design.

The online bank accounts opening process designed using BP4Cloud has the same control flow elements (NOAC), activities elements (NOA, control paths (MCC) (in our use case *e*=15 and *n*=15) and the control-flows (CFC). The presented metrics clearly demonstrate that using BP4Cloud is not in any view more complex. In fact, BP4Cloud does neither modify the control-flow elements of the BP nor its activities.

Activity-Service Matching Algorithm Evaluation

To evaluate the effectiveness of our algorithm, experiments were conducted to evaluate its precision and recall on the BP presented in the motivating scenario section. The precision evaluates the capability of the algorithm to retrieve top-ranked services that are most relevant to the activity. The recall evaluates the capability of the algorithm to get all the relevant CSs [Davis and Goadrich, 2006].
Formally:

$$P = \frac{|S_{Rel}|}{|S_{Ret}|}, \qquad R = \frac{|S_{Rel}|}{|Rel|}$$

$$P_k = \frac{S_{Rel,k}}{k}, \qquad P_r = \frac{S_{Rel,|Rel|}}{|Rel|}$$

where Rel denotes the set of relevant service, S_{Ret} is the set of retuned services, S_{Rel} is the set of returned relevant services and $S_{Rel,k}$ is the set of relevant services in the top k returned services. Among the above metrics, P_r is considered to most precisely capture the precision and ranking quality of the algorithm. We also plotted the recall/precision curve (R-P curve). In an R-P curve figure, the X-axis represents recall, and the Y-axis represents precision. An ideal discovery result presents a horizontal curve with a high precision value. The R-P curve is considered by the IR community as the most informative graph showing the effectiveness of the discovery algorithm [Davis and Goadrich, 2006].

To evaluate our proposed algorithm, we asked several BP designers and technical developers who are familiar with CSs uses to identify the relevant services meeting the business activities. Table 1 illustrates the online bank account process results. For reason of space constraint, we display some relevant services related to the activities ”check customer summary” and ” send account id”.

We evaluated the precision of the proposed services and report the average top-2, top-5, and top-10 precision. To ensure the top-10 precision is meaningful, we selected activities for which we can identify more than 15 relevant services over different providers. Figure 7 illustrates the results. The top- 2, top-5, and top-10 precisions related to the context matching of our algorithm are respectively 91%, 87%, 74%. The precision related to the functional matching can be 1 or 0. Finding the suitable operation performing the activity is considered as 1 in terms of precision value, else the precision is estimated to 0.

The evaluation demonstrates that taking into account the context of both the business process and activity using their business features during the context matching can effectively provide acceptable precision values (91%, 87%), which means, we can identify suitable candidate services. The discovery of suitable service operation can more challenging since the functional matching involves different matching level. The inputs/outputs matching is considered as the main important step to validate if a candidate operation can perform the activity.

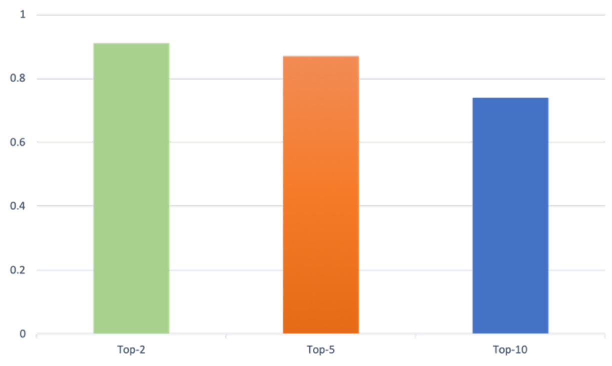


Fig 7. Top-k precision for retrieved services.

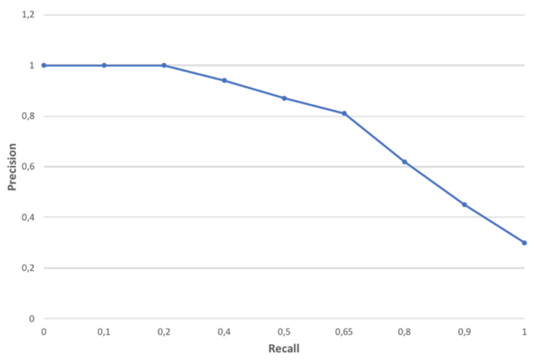
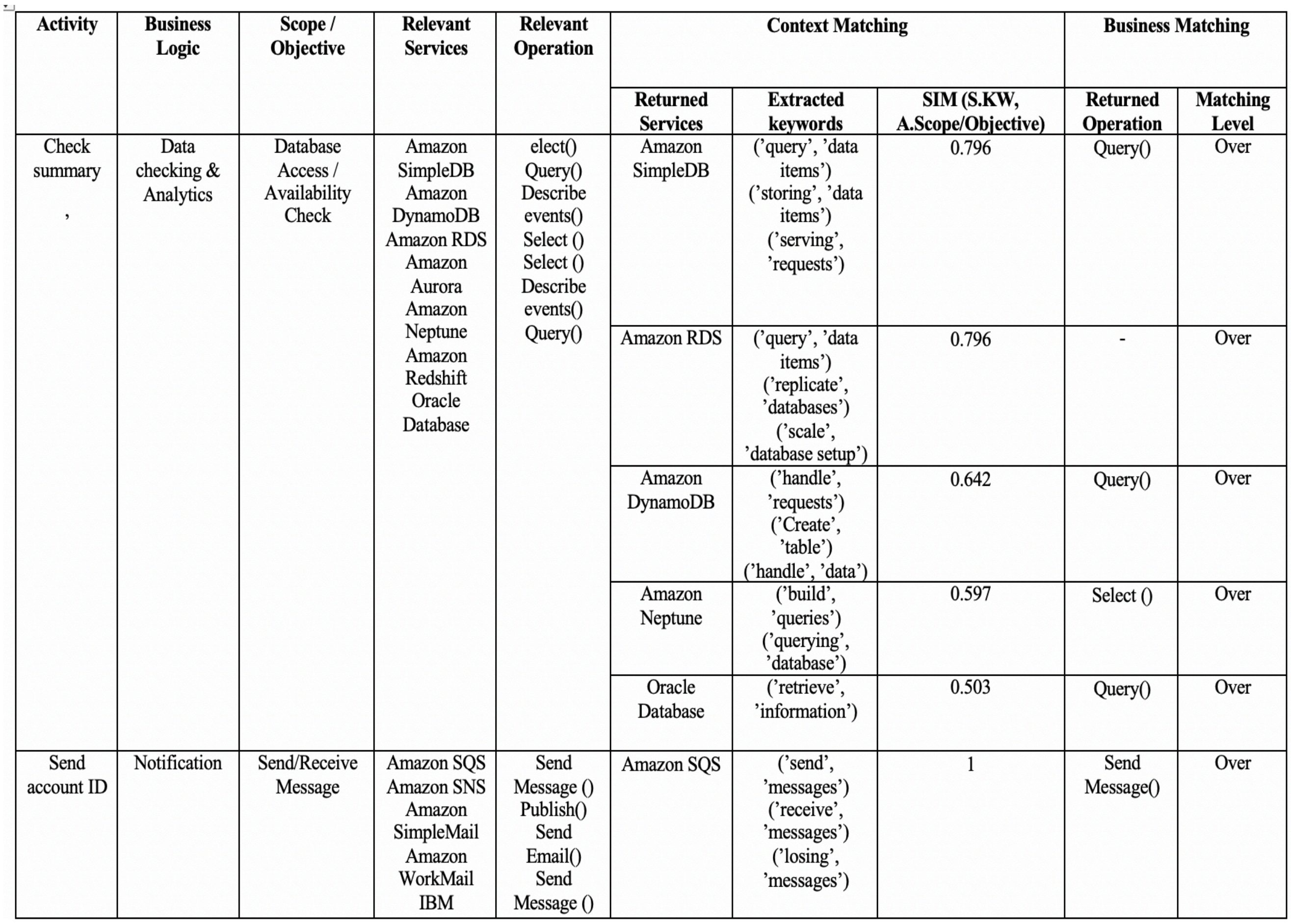


Fig 8. R-P Curves

We plot the average R-P curves to illustrate the overall performance of the matching algorithm. As mentioned previously, an appropriate discovery result has a horizontal curve with a high precision value. Typically, precision and recall are inversely related, i.e., as precision increases, recall falls and vice-versa. A balance between these two needs to be achieved. As illustrated by Figure 8, for a recall average equals to 0,63 we have 0,85 as precision value. As an example, for the service activity "Check customer summary", we have 20 services considered as relevant in ULIT i.e., $|Rel| = 20$, the matching algorithm returns a total of 16 services i.e $|S_{Ret}| = 16$, among them 13 services are considered relevant i.e., $|S_{Rel}| = 12$. We obtain a precision value $P = 13/16 = 0, 81$ and a recall value $R = 13/20 = 0,65$.

In some cases, depending on particular context, high precision at the cost of a recall or high recall with lower precision can be chosen. Thus, evaluating a matching algorithm for services discovery must be related to the purpose of the discovery. In our case a compromise between the recall and the precision values is necessary. Therefore, we can announce the proposed algorithm provides accurate results for CSs discovery.

TABLE I. Activity-Services matching algorithm evaluation



Activity	Business Logic	Scope / Objective	Relevant Services	Relevant Operation	Context Matching			Business Matching	
					Returned Services	Extracted keywords	SIM (S.KW, A.Scope/Objective)	Returned Operation	Matching Level
Check summary ,	Data checking & Analytics	Database Access / Availability Check	Amazon SimpleDB Amazon DynamoDB Amazon RDS Amazon Aurora Amazon Neptune Amazon Redshift Oracle Database	elect() Query() Describe events() Select () Select () Describe events() Query()	Amazon SimpleDB	('query', 'data items') ('storing', 'data items') ('serving', 'requests')	0.796	Query()	Over
					Amazon RDS	('query', 'data items') ('replicate', 'databases') ('scale', 'database setup')	0.796	-	Over
					Amazon DynamoDB	('handle', 'requests') ('Create', 'table') ('handle', 'data')	0.642	Query()	Over
					Amazon Neptune	('build', 'queries') ('querying', 'database')	0.597	Select ()	Over
					Oracle Database	('retrieve', 'information')	0.503	Query()	Over
Send account ID	Notification	Send/Receive Message	Amazon SQS Amazon SNS Amazon SimpleMail Amazon WorkMail IBM	Send Message () Publish() Send Email() Send Message ()	Amazon SQS	('send', 'messages') ('receive', 'messages') ('losing', 'messages')	1	Send Message()	Over

Related Work

Despite various efforts for cloud resource integration in BPs modeling, it remained poorly described and operated [Nacer et al., 2019]. In [Ramos-Merino et al., 2019], the authors proposed extensions to support new features such as the possibility to describe the workflow behavior, the activities performed and the context of the application in a machine understandable way. Even though, this works can simplify the BPs modeling, it did not take into consideration several resources required for the development and execution of the BP, particularly, cloud resources perspective. In [Heidari et al., 2013], the authors proposed the use of BPSim, which is a standard that provides a framework for structural and capacity analysis of BP models specified by the use of BPMN or XPDL (XML Process Definition Language). However, it is limited to introduce BPMN extensions to enhance its expressive capabilities without considering the runtime environment. In fact, the resource perspective may change depending

on the runtime environment, notably, cloud environments require specific resources that are different from other runtime environments.
Other approaches investigated the discovery of concrete CSs performing BP's activities. In that respect, several semantic approaches are interesting such as [Martino et al., 2018], [Nagara-jan et al., 2018]. However, most of these works are depending on the preexistence of providers' specific ontologies (OWL-S services description) that require mapping techniques to coordinate the difference between agnostic (abstract) and vendor dependent concepts to support interoperability. Moreover, from the business designer perspective, it requires to have intimate knowledge of semantic services and related description and implementation. The business designers may not be aware of all the knowledge that constitutes the domain ontology. As a result of which many relevant services may not be considered in the service discovery process. Therefore, proposing an atomic mapping from business activity to CSs based on available services description that does not make any assumptions about the description standard languages of CSs, presents valuable insights to improve BP development.

Conclusion

This paper provides efficient support to discover cloud services required for business process development. We define a cloud-aware BP by proposing BP4Cloud which is a BPMN extension that supports the design of cloud resource perspective requirements. BP4Cloud offers a solution for coordinating cloud resources between business designers and technical developers. Following the modeling of the required cloud resource, we propose an Activity-Services Matching algorithm to assist technical developers in discovering the required cloud services. Our proposed algorithm is conducted through two steps: the context matching which discovers services that share the same business context as the business activity and the business matching that identifies the suitable operation performing the activity. Our experimental evaluation demonstrates that the Activity-Services Matching algorithm can assist technical developers owing to its precision. Although we believe that our algorithm leaves scope for a range of enhancements, yet it provides suitable results. As ongoing work, we intend to conduct the composition of the discovered cloud resources to develop the cloud workflow application.

References

Carrillo, A. and Sobrevilla, M. (2017). Bpm in the cloud: A systematic literature review. In Software Engineering (cs.SE).

Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In International conference on Machine learning, pages 233–240.

Elsevier Mendeley Data (2019). Elsevier Mendeley Data . URL: https://www.elsevier.com/ authors/author-services/research-data/ mendeley-data-for-journals [accessed: 30-01- 2020].

Gabsi, H., Drira, R., and Ghezala, H. H. B. (2018). Personalized iaas services selection based on multi-criteria decision making approach and recommender systems. In International Conference on Internet and Web Applications and Services, pages 5–12. ISBN: 978-1- 61208-651-4.

Gabsi, H., Drira, R., and Ghezala, H. H. B. (2019). A hybrid approach for personalized and optimized iaas services selection. In International Journal on Advances in Intelligent Systems.

Geigera, M., Harrera, S., Lenhardb, J., and Wirtz, G. (2018). Bpmn 2.0: The state of support and implementation. In Future Generation Computer Systems, volume 80, pages 250–262.

Heidari, F., Loucopoulo, P., and Frances Brazier, J. B. (2013). A meta-meta-model for seven business process modeling languages. In IEEE Conference on Business Informatics, pages 216–221.

Huber, J., Polan, G., Mateja, and Jot, K. (2018). Towards the component-based approach for evaluating process diagram complexity. In International Symposium on Business Modeling and Software Design, volume 11, pages 260–269.

Lin, D. (1998a). An information-theoretic definition of similarity. In International Conference on Machine Learning, volume 1, pages 296–304.

Lin, D. (1998b). An information-theoretic definition of similarity. In 15th International Conference on Machine Learning, pages 296–304.

Lucineia, T., Manfred, R., and Iochpe (2009). Activity patterns in process-aware information systems: basic concepts and empirical evidence. In International Journal of Business Process Integration and Manage- ment, volume 4, pages 93–110.

Marneffe, M.-C. and Manning, C. D. (2015). The stanford typed dependencies representation. In Proceedings of the workshop on Cross-Framework and Cross- Domain Parser Evaluation, pages 1–8.

Martino, B. D., Esposito, A., Nacchia, S., and Maisto, S. A. (2017). A semantic model for business process patterns to support cloud deployment. In Computer Science - Research and Development, volume 32, pages 257–267.

Martino, B. D., Pascarella, J., Nacchia, S., Maisto, S. A., Iannucci, P., and Cerr, F. (2018). Cloud services categories identification from requirements specifications. In International Conference on Advanced Information Networking and Applications Workshop, volume 1, pages 439–441.

Nacer, A. A., Godart, C., Rosinosky, G., Taria, A., and SamirYoucef (2019). Business process outsourcing to the cloud: Balancing costs with security risks. In Computers in Industry, volume 104, pages 59–74.

Nagarajan, R., Thirunavukarasu, R., and Selvamuthukumaran (2018). Cloud broker framework for infrastructure service discovery using semantic network. In International Journal of Intelligent Engineering and Systems, volume 11, pages 11–19.

Niwattanakul, S., Singthongchai, J., Naenudorn, E., and Wanapu, S. (2013). Using of jaccard coefficient for keywords similarity. In Proceedings of the International MultiConference of Engineers and Computer Scientists, volume 1, pages 296–304.

Ramos-Merino, M., Santos-Gago, J. M., lvarez Sabucedo, L. M., Alonso-Roris, V. M., and Sanz-Valero, J. (2019). Bpmn-e2: a bpmn extension for an enhanced workflow description. In Software and Systems Modeling, volume 18, pages 2399–2419.

Sun, C., Wang, Z., Wang, K., Xue, T., and Aiello, M. (2019). Adaptive bpel service compositions via variability management: A methodology and supporting platform. In International Journal of Web Services Research, volume 16.

TRAN, V. X., PUNTHEERANURAK, S., and TSUJI, H. (2009). A new service matching definition and algorithm with sawsdl. In IEEE International Conference on Digital Ecosystems and Technologies, volume 1, pages 371–376.

ULIT (2019). ULIT . URL: http://dx.doi.org/10. 17632/7cy9zb9wtp.2 [accessed: 30-01-2020].

Workflow Resource Patterns (2018). Workflow Resource Patterns. URL: http://www.workflowpatterns. com/patterns/resource/ [accessed: 30-01-2020].