Towards Business-to-IT Alignment in the Cloud

Kyriakos Kritikos[1(✉)], Emanuele Laurenzi[2], and Knut Hinkelmann[2]

[1] ICS-FORTH, Heraklion, Greece
kritikos@ics.forth.gr
[2] FHNW University of Applied Sciences and Arts Northwestern Switzerland, Olten, Switzerland
{emanuele.laurenzi,knut.hinkelmann}@fhnw.ch

Abstract. Cloud computing offers a great opportunity for business process (BP) flexibility, adaptability and reduced costs. This leads to realising the notion of business process as a service (BPaaS), i.e., BPs offered on-demand in the cloud. This paper introduces a novel architecture focusing on BPaaS design that includes the integration of existing state-of-the-art components as well as new ones which take the form of a business and a syntactic matchmaker. The end result is an environment enabling to transform domain-specific BPs into executable workflows which can then be made deployable in the cloud so as to become real BPaaSes.

Keywords: BPaaS · Service · Design · Discovery · Selection
Alignment · Mediation

1 Introduction

Due to intense market competition, organisations can survive only if they offer services that are either innovative or exhibit a better quality than their competitors. However, by owning a limited infrastructure and continuously requiring to improve the existing business processes (BPs) leads to reaching certain impassable limits. Moreover, the infrastructure maintenance, operation and management costs can be quite prohibiting, especially for small or medium enterprises.

Fortunately, cloud computing can become the medium via which organisations can acquire cheap, commodity resources on-demand while also being able to achieve certain benefits, including: outsourcing infrastructure management with reduced cost, flexible resource management, and elasticity. Such benefits can certainly enable improving and optimally controlling BP performance.

However, as cloud computing handles only the infrastructure level, an organisation now faces the hard and yet unsolvable problem of aligning the business with the IT level. Moreover, many organisations do not have the expertise and know-how to use and combine the cloud services offered.

The above problems can be solved by combining BP management with cloud computing to realise the BP as a service (BPaaS) paradigm to enable migrating and more optimally managing BPs in the Cloud [6,31,32]. However, such a

Z. Á. Mann and V. Stolz (Eds.): ESOCC 2017, CCIS 824, pp. 35–52, 2018.
https://doi.org/10.1007/978-3-319-79090-9_3

combination is not trivial as it leads to the following challenges which especially concern the BP design lifecycle activity: (a) how to map a BP to a technical workflow with a suitable automation level; (b) how to align business terms and requirements with technical ones to drive the selection of the most suitable services to be then integrated into the workflow; (c) how to deal with the service incompatibility problem effectively to guarantee the correct execution of the designed workflow. Such a problem relates to checking the syntactic compatibility of messages exchanged between two or more selected workflow services.

To realise the vision of BPaaS, the CloudSocket project (www.cloudsocket.eu) delivers a platform that unifies together environments supporting different BP lifecycle activities. This paper presents our contribution in form of a BPaaS Design Environment able to deal successfully with all aforementioned challenges. This translates to introducing an innovative architecture with suitable components that support: smart and semantic service discovery at both business and technical levels, optimal cross-level service selection, mapping between business and technical requirements and mediation between the execution of two or more services to achieve message-level compatibility. In result, the developed environment enables a BPaaS provider to transform the initial business functional and non-functional requirements that match the necessities of potential BPaaS customers into an executable workflow. That workflow can then become deployable in the cloud by using other CloudSocket environments.

The BPaaS Design Environment was built by exploiting state-of-the-art as well as two novel components. The first component, the business matchmaker, enables to find services that satisfy the user functional and non-functional requirements at the business level by following a novel questionnaire-based approach. Such services are then filtered and selected by employing state-of-the-art technical service matchmaking and selection components. Service selection relies on the second novel component, the syntactic matchmaking one, able to infer the message-based compatibility between two or more selected services and produce a mapping specification. This specification can then be exploited by a service mediation service to support the compatible message transformation between services and thus guarantee the smooth operation of the BPaaS workflow in which this mediation service is integrated.

This paper is structured as follows. Section 2 shortly analyses existing research results, some of which are exploited in the production of the BPaaS Design Environment. Section 3 analyses the environment's main architecture by also explaining the main functionality and role of its components. Sections 4 and 5 detail the architecture's two main novel components, the business and syntactic matchmakers. Section 6 introduces a use case to demonstrate the main benefits of the proposed environment and to validate it. Finally, the last section concludes the paper and draws directions for further research.

2 Background

2.1 Business-to-IT Alignment

Business-to-IT alignment typically refers to the gap between business requirements and technical solutions [12]. Cloud offerings are technically described making it hard for business people to properly assess the best fitting cloud solution [33]. Thus, identifying suitable cloud solutions requires specifying requirements for and capabilities of a service in both a business and IT language. To ensure knowledge understandability and transparency, it is a common practice to represent knowledge in models [11,29]. Models abstract away from complex realities and achieve precise modelling of the intended domain. In [13] we already adopted a model-driven approach where an extension of BPMN 2.0 allows modeling both BP requirements for business and workflows/cloud services in a technical language. That approach includes translating the business to the technical language to enable matching process requirements and workflow/cloud service capabilities. Translation and matching are performed by semantically lifting models with ontologies to make them machine-interpretable.

[2] defines *Semantic Lifting* as "the process of associating content items with suitable semantic objects as metadata to turn unstructured content items into semantic knowledge resources". Semantic Lifting shifts the purpose of modelling beyond transparency and communication [14]. The interpretable knowledge base (ontology) allows reaching higher system automation levels based on models [13]. For example, an ontology-based early warning system assessing supply chain risks was proposed in [8], while in [7] ontologies are combined with a case-based reasoning approach to support workplace learning. Closer to our current problem, [9] introduced the AML ontology for automatic identification of correspondences between BP model activities. Similar BP matching approaches are described in [1]. Such approaches are not sufficient for BP-to-workflow matching as a BP is far less detailed than a workflow such that a BP activity is most likely to refer to a whole workflow fragment. As such, due to this different degree of detail between the two levels, such approaches suffer from inaccurate matching, something not only addressed but also far improved by our approach.

Approach. We follow a model-driven which performs domain-specific conceptualization (mapping to well-known benefits [10,17,25]) on two levels, where the one targets BP users, while the other targets IT service experts. This allows designing domain-specific models capturing suitable domain knowledge on both levels. This approach builds upon the findings in [13] but adopts a different perspective on business-to-IT alignment in the Cloud. Namely, there is a shift from language translation to the mapping of values between requirements and specifications on both the business and IT levels, separately. Hence, the Business-IT alignment paradigm is applied sequentially by further refining results from the business to the IT level. As such, 3 matchmaking components are proposed: (a) the business and (b) technical matchmakers enhanced with formal semantics for machine-interpretation plus (c) the syntactic matchmaker. The combination of these 3 components allows identifying the most suitable cloud services within both business and technical terms that will eventually form a workflow.

2.2 Technical Service Matchmaking

Technical service matchmaking involves functional and QoS matching. Functional matching usually focuses on I/O-based matching [18,26] while QoS matching takes the view of QoS as conformance [20] and employs different kinds of techniques [21] to infer if the service's solution space is included in that of the request. While most work focuses on one aspect individually, some approaches consider both aspects simultaneously [3,16]. However, they usually sequentially combine the matching in both aspects and do not employ semantic techniques, thus not exhibiting the right performance and accuracy level.

As such, our previous work [23] explored different ways the 2 matching types can be jointly performed: (a) sequential combination; (b) parallel combination; (c) subsumes-based combination. The experimental evaluation of these combinations showed that the parallel one leads to the best possible results with respect to performance, as matchmaking accuracy is perfect in all combinations.

Our approach exploits two aspect-specific matchmakers, a functional and a non-functional. The functional is a state-of-the-art matchmaker developed in the Alive project [4] which relies on the combination of I/O-based and IR-based matching. It exploits a smart graph-based structure to dynamically tolerate changes in domain ontologies (i.e., the ontologies via service I/O is annotated) as well as supply almost constant-in-time query operations over the graph.

The unary matchmaker [21] follows a hybrid QoS service matching approach. First, it aligns ontology-based service specifications based on their QoS terms. Then, it performs service filtering in a step-wise manner by considering each QoS term individually in each step. As unary constraints are assumed to be involved in service offers and demands, the matchmaker employs smart structures to support term-based filtering which results in ultra fast matching time.

2.3 Service Selection

Service selection work usually considers only one abstraction level by also neglecting semantics, thus producing results of imperfect accuracy. Accuracy is further reduced as some algorithms employ smart but non-optimal solving techniques, like Genetic Algorithms to accelerate the service selection time.

As service selection for a BPaaS includes different abstraction levels, we have developed a cross-level constraint-based algorithm [22] which exhibits the following features: (a) handling of multiple optimisation objectives by employing the Analytic Hierarchy Process (AHP) [27] and Simple Additive Weighting (SAW) [15] techniques; (b) the capability to bridge the gap between the two levels (SaaS and IaaS) via inserting functions that derive the QoS at the SaaS level based on the capabilities selected at the IaaS level; (c) the addressing of overconstrained requirements by employing smart utility functions that allow slightly violating these requirements so as to produce at least one solution; (d) consideration of dependencies between QoS parameters at the same level enabling a more accurate evaluation of respective solutions; (e) the capability [19] to accelerate solving time by fixing parts of the problem to certain partial solutions by relying on the BPaaS execution history.

State-of-the-Art Advancement. The proposed BPaaS Design Environment advances the state-of-the-art by exhibiting an innovative combination of existing, like holistic technical service matchmaking, and new features. The innovative business matchmaker follows a dynamic questionnaire-based approach enabling business users to answer a minimum set of questions before the mapping of the designed BP to a set of services, able to realise its functionality, can be produced. Such an approach is more natural and user-intuitive as it employs questions mapped to a natural language with terms drawn from the business domain. It also supports producing a minimal set of services to be further filtered and selected based on technical requirements such that the solution space is significantly reduced and service discovery time accelerated.

The novel syntactic matchmaker enables producing a correct executable workflow via the suitable integration of services at the technical level based on their message compatibility. Such compatibility is guaranteed by generating mapping specifications that are exploited by mediation tasks incorporated in the generated workflow. Finally, our framework addresses all layers involved in a BPaaS system along with their dependencies thus being able to produce a more complete and optimal BPaaS design product.

3 Architecture

The creation of the BPaaS Design Environment was underpinned by the design science research (DSR) methodology in [30]. First, the literature on Business-IT alignment in the Cloud was screened. Then, CloudSocket created the settings to contribute to the problem awareness: application scenarios were created in workshops involving both industrial and scientific experts. The results and insights were useful to suggest the BPaaS Design Environment's first draft which was then finalised in a web-based solution through continuous development. Finally, as shown in Sect. 6, the validation took place with respect to the most agreed application scenario among the members of the CloudSocket consortium.

The BPaaS Design Environment follows a model-driven and semantics-aware approach for business-to-IT alignment in the cloud which comprises 3 main transformation steps: (a) BP-to-business-services; (b) business-services-to-technical-services; (c) BP & technical-services-to-executable-workflow. The approach guarantees the produced solution's technical feasibility by employing a two-step service matchmaking process at both business and technical levels and a service selection algorithm that is syntactic-compatibility-aware at the technical level.

To achieve its main goal, the environment exhibits an architecture, depicted in Fig. 1, comprising 8 main components that are now analysed in detail. Some components correspond directly to some of the aforementioned steps while others play a supporting or orchestration role.

BPaaS Designer (BD). It represents the main point of interaction with the user during BPaaS design. It enables specifying both domain-specific BPs and executable workflows. It also guides users in providing suitable input to support the BP-to-workflow alignment.

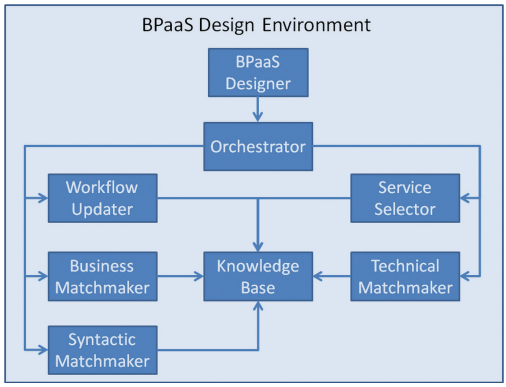


Fig. 1. The architecture of the BPaaS Design Environment

Orchestrator (Orch). Orchestrates its underlying components to handle requests issued by the BPaaS Designer.

Business Matchmaker (BM). Matches the cloud services registered in the Knowledge Base based on business requirements derived from a questionnaire-based approach explained in Sect. 4.

Technical Matchmaker (TM). It exploits technical state-of-the-art aspect-specific matchmakers in a parallelised fashion according to the approach in [23].

Service Selector (SS). It [22] produces a concrete optimal solution for the service-based workflow at hand by considering the user technical non-functional requirements while also attempting to maximise the message compatibility between services by exploiting the next component.

Syntactic Matchmaker (SM). Called dynamically by the SS while solving the service selection problem to find the message compatibility [24] between the next and all previously selected services in each BPaaS workflow's execution path where such a service participates. When an incompatible solution is constructed, SS can backtrack and check another one. To smartly deal with cases where the same call is issued, e.g., due to deep backtracking, SM stores the call results to immediately answer it. The mapping of the output parameters to the input ones of the next service is also recorded to enable updating the BPaaS workflow via a mediation service, as performed by the next component.

Workflow Updater (WU). Updates the BPaaS workflow by performing the following actions for each workflow's execution path: (a) replays the solution construction in each path to obtain the mapping of the current service in the path from the SM; (b) introduces a mediation service within the workflow, immediately before the current service, which takes as input the current output parameter set and the mapping specification and produces as output the input parameters of the current service.

Knowledge Base (KB). Includes all necessary and sufficient information to support all reasoning/matching/selection tasks executed in the system.

4 Business Matchmaking

The *Business Matchmaker* allows specifying requirements in a more user-centric approach than that in [13]. It relies on a context-adaptive questionnaire that guides the user via a set of questions reflecting BP functional and non-functional requirements. Follow-up questions are displayed based on the result of a prioritisation algorithm that considers: (a) user preferences in terms of categories (e.g., *Performance* rather than *Data Security*); (b) information value (or entropy) of semantic attributes reflecting cloud service specifications at the business level, e.g., how distinguishing an attribute, such as *monthly downtime*, is for service filtering. Namely, the higher the entropy value of an attribute, the higher its service distinguishability degree, and thus the higher the assigned priority of the related question. This approach leads to the least possible number of questions being answered, thus reducing the business service matching time. The idea is that the questionnaire can be applied on the whole BP first. If no service is found, we then move down to groups of activities, until the level of single activities.

4.1 The Context-Adaptive Questionnaire

The Context-Adaptive Questionnaire relies on our BPaaS ontology [11]. Questions focus first on functional requirements and then on non-functional ones. The questionnaire enables the user to specify functional requirements in two ways by:

- inserting an action and object from a predefined taxonomy in the BPaaS ontology. This corresponds to the convention of BPMN to name activities by a verb (i.e., action) and a noun (object) [28] whose combination provides the "what-is-about" knowledge.
- inserting the most suitable category from APQC Process Classification Framework.

Next, the user can choose one of the 5 non-functional (NF) categories: *Data Security*, *Payment*, *Performance*, *Service support*, and *Target Market*.

The NF categories were derived from the Cloud Service Agreement Standardisation Guidelines [5], published by EC to standardize and streamline the terminologies and understanding of cloud services. The NF categories were subsequently discussed and validated within the CloudSocket consortium. In result, a set of questions and sub-questions were derived out of them. For instance, the *Performance* category includes questions like the following:

- What is your preferred monthly downtime in minutes?
  Possible answer: 30 min
- Should the process be executed on a daily, weekly, monthly or yearly basis?
  Possible answer: On a weekly basis

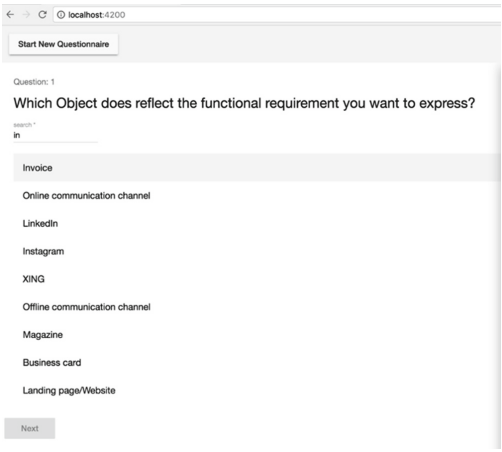


Fig. 2. The object selection for the functional requirements posing

– What is your favorite response time level?
  Possible answer: High, Medium or Low
– How many simultaneous users should the cloud service support?
  Possible answer: at most 10

For each question, we have distinguished among 4 types of answers as: (1) single-answer selection; (2) multi-answer selection; (3) search-insert; (4) value-insert. Value- and search-insert require user input. While the former enables inserting attribute values (e.g., the aforementioned downtime), the latter enables crawling predefined values from the ontology and selecting the suitable one. For instance, answers related to the first 3 functional requirement questions (Action, Object and APQC category) are of search-insert type. Namely, users can insert keywords for the BP they are looking for, and the ontology returns the concepts matching these keywords. Figure 2 shows this functionality's implementation result.

Each time a question is answered, semantic rules are applied to convert implicit knowledge reflecting the business requirements into an explicit one. This prepares the ground to identify matching cloud services by applying a semantic query. For example, assume we have the following:
Specifications from the KB as follows:

– A cloud service with the execution constraint of 20 times per day.

Requirements from the questionnaire as follows:

- Should the process be executed on a daily, weekly, monthly or yearly basis?
  Answer: At least on a weekly basis.
  - How many times should the process be executed?
    Answer: At least 10 times

Running a process at least on a weekly basis implies that can also run on a daily basis. The semantic rule, therefore, would infer the answer "On a daily basis" and insert it in the KB. The semantic query then compares the derived fact with the cloud service fact related to the execution constraint. In result, the cloud service specification matches with the requirement.

4.2 Question Prioritisation Algorithm

The NFR questions follow a question prioritisation algorithm. This enables identifying the matching cloud services by asking as few questions as possible. Answers to the questions, along with previous ones, are used to display the follow-up question. The algorithm considers the following:

- Grouping among non-functional attributes. For instance, if the user selects to answer one from *availability* and *response time* attributes of the *Performance* category, the follow-up question will be on the other attribute in this category.
- Entropy expressing the variation degree in the values of each non-functional attribute. Entropy of an attribute is "0" when every cloud service stored in the *KB* contains the same attribute value, while "1" in the opposite case.

The entropy formula is expressed as follows:

$$Entropy\,(attr_i) = -\sum_{j=1}^{J} (p_{ij} \cdot log_2\,(p_{ij}))$$

where J is the total number of attribute values and p_{ij} is the probability that a certain attribute value val_{ij} of attribute $attr_i$ appears in a certain cloud service. As this probability can be regarded as independent and uniform across all attribute values, p_{ij} can be expressed as: $p_{ij} = \frac{[CS]_{csval_{ik}=val_{ij}}}{[CS]}$ where the nominator denotes the number of cloud services that exhibit the respective attribute value ($csval_{ik}$ denotes the value of $attr_i$ for cloud service k) and the denominator the number of all services.

The prioritisation algorithm's signature and main logic is as follows.
Input.

- Already stated variables: $attr, CS, val, csval$.
- The set of non-functional categories C ={*Data Security*, *Payment*, *Performance*, *Service support*, *Target Market*}.
- Set of tuples $< attr_i, Q_l >$ where Q is the set of questions and Q_l is a certain question where $1 \leq l \leq [Q]$. So, each tuple maps 1 attribute to 1 question.

Output. The filtered set of cloud services CS that match with the content of the questionnaire, i.e., questions and answers.
Business Logic.

1. IF the number of categories left is positive ($|C| > 0$), select a category c_n, ELSE exit.

2. IF c_n has a positive number of semantic attributes left, i.e., $|attr_i$ s.t $attr_i.cat = c_n| > 0$, THEN calculate the entropy of all the selected category's attributes, ELSE remove the current category c_n from C and go to (1).
3. Select attribute $attr_i$ with highest entropy.
4. Display question Q_l that is mapped with the $attr_i$.
5. Get user answer mapping to a value val_{ij} of attribute $attr_i$.
6. Filter services in CS which do not satisfy the condition: $csval_{ik} = val_{ij}$.
7. Remove the semantic attribute $attr_i$ from the category c_n and go to (2).
8. Exit.

5 Syntactic Matchmaking

Business/technical matching cannot guarantee the message compatibility between selected services in a BPaaS workflow. Such a compatibility is thus a hard constraint in service selection for producing optimal, message-compatible solutions that can be safely executed. As such, the TM was developed to derive such compatibility and offer it as a function to SS.

The main idea is that the TM should first find which output messages of previously selected services match to which input messages of the currently selected service (based on SS's solution generation process) for each execution path in the BPaaS workflow. Then, it should check for each message-to-message match if the first message conveys less information than that required by the second message. If this checking succeeds, no compatibility between the execution path's considered services exists. When all message pair matches are compatible, the considered services are message-compatible.

Message Matching. The first message compatibility step can rely on existing semantic service annotations to easily and rapidly discover matching message pairs, as the messages involved in these pairs should map to semantically compatible concepts. However, even in the presence of such knowledge, message matching is not trivial and follows a two-step process involving semantic & syntactic message matching. This process is exemplified via the example of a certain service pair involving service S_2 with 2 input parameters mapped to ontology concepts A & B and service S_1 with 2 output parameters mapped to ontology concepts C & D.

At the semantic level, a bipartite matching approach is followed checking whether every parameter of the current service has a mapping to one parameter of the previously selected services (or the initial user input) in a certain execution path and attempting to discover a solution with the lowest overall distance. As such, we first define a local matching degree between two parameters to be the distance between the parameters' annotation concepts in the ontology subsumption hierarchy, provided that the second parameter's concept subsumes the first parameter's one. If the latter does not hold, the distance is infinite. This guarantees that no information loss occurs as in the opposite case, the more concrete concept in the S_2 input will require specifying additional pieces

of information than those exhibited in the concept in the S_1 output. A mapping solution's overall distance is then the sum of the distances of the matches found. As such, the matching problem can be defined as follows:

$$min \left\{ \begin{array}{l} \frac{1}{[J]} \cdot \left(\sum_{i \in I}^{j \in J} \left(\frac{dist(M_i, N_j)}{maxPSize} \cdot x_{ij} \right) + \sum_{j \in J} \left(1 - \sum_{i \in I} x_{ij} \right) \right) \\ \sum_{j \in J} x_{ij} \leq 1 \\ \sum_{i \in I} x_{ij} \leq 1 \\ i = [1, \ldots, [I]], j = [1, \ldots, [J]] \end{array} \right\}$$

where I and J are the sets of input and output parameters, respectively, x_{ij} is a decision variable whether the output parameter i matches the input one j, $dist\,(M_i, N_j)$ is the distance between annotation concepts M_i and N_j of the two parameters pair while $maxPSize$ represents the maximum subsumption path length in the respective domain ontology used.

Suppose that the following relations hold in the running example: A subsumes B, C & B, C subsume D. In this respect, the best possible matching is $\{A \rightarrow C, B \rightarrow D\}$ with overall distance of 2. The other matching solution $\{A \rightarrow D, B \rightarrow C\}$ is not selected as the local distance between B & C is infinite so the overall distance is also infinite.

The algorithm then proceeds at the syntactic level by considering only those message pairs with a finite local degree of match. For each message pair filtered, we note the information items for the output parameter and those of the input parameter and then we check whether the former include the latter. As the information items have been already matched to ontology concepts, we perform this checking by replacing the information items with the attributes of the ontology concept. Even if the concepts matched are not identical, as they are related with a subsumption relation, they will have common attributes. So, the problem then is mapped to checking whether the concept attributes of the output parameter form a superset of those of the input parameter.

Message types might also convey information not included in an ontology requiring to perform a different matching kind for them. This matching's logic is similar to that for the semantic level. In particular, bipartite matching is performed with the exception of how the distance is calculated at the local level. At that level, we consider both how similar the field names are and how close are their types. Name similarity can rely on well-known string distance measures (e.g., Levenshtein) while type similarity relies on the approach in [24] mapping to the compatibility level between types. The local overall distance would then equal the weighted sum of the two different distances.

If all input parameter parts are matched, the compared messages are semantically compatible. Otherwise, the compared services are semantically incompatible.

Let us continue the running example to explain syntactic matchmaking. Suppose that A & C were found equivalent. C maps to message type MT_1 containing 4 information pieces MT_{11}, MT_{12}, MT_{13} and MT_{14}. A maps to message type MT_2 containing 3 information pieces MT_{21}, MT_{22}, and MT_{23}.

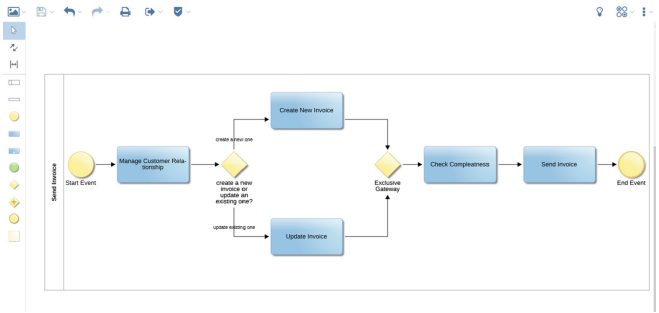


Fig. 3. The Send Invoice business process in BPMN 2.0

Based on matching message types to ontology concepts, we have that MT_{11} and MT_{21} map to $A.A1$ while MT_{12} and MT_{22} map to $A.A2$. Thus, the information pieces are transformed into $\{A.A1, A.A2, MT_{13}, MT_{14}\}$ for first message type and $\{A.A1, A.A2, MT_{23}\}$ for the second. For those pieces not mapping to ontology attributes, we solve a bipartite matching problem again. Suppose that $dist\,(MT_{13}, MT_{23}) = 0.8$ and $dist\,(MT_{14}, MT_{23}) = 0.2$. Then, the sole mapping to be selected will be $\{MT_{13} \rightarrow MT_{23}\}$. If we replace MT_{23} with MT_{13}, we then need to check whether $\{A.A1, A.A2, MT_{13}\}$ is subset of $\{A.A1, A.A2, MT_{13}, MT_{14}\}$ which holds.

6 Validation

Our approach was validated based on a use case developed by CloudSocket's industrial partners. We focused on a very common BPs among SMEs - the Send Invoice one. This BP is modelled in BPMN, see Fig. 3, via our BPaaS Design environment. It starts with the "Manage Customer Relationship" activity; next an exclusive gateway splits the BP flow between either creating a new invoice or updating an existing one. Then, invoice completeness is checked, and finally the invoice is sent. Subsequently, starting with this BPMN process, we acquaint the reader with a prerequisite plus the main steps involved in our approach.

Prerequisite Step: Service Profile Registration. The following services were inserted in the *KB* as instances of *CloudService* class:

- YMENS, Zoho and Sugar CRM were inserted as CRM systems which were annotated with the action *Manage*, the object *Customer* and the APQC category *3.5.2.4 Manage Customer Relationship*
- Mathema Document Generator, Open Source Billing, Simple Invoice and InvoiceNinja as invoicing systems annotated with action *Generate*, object *Invoice* and APQC category *9.2.2.2 Generate Customer Billing Data*
- Gmail, Ninja_email and Mailjet were inserted as e-mail systems which were annotated with the action *Manage*, the object *Invoice* and the APQC category *9.2.2.3 Transmitting Billing Data to Customers*

Table 1 shows a part of the non-functional profiles of the considered services.

Table 1. Functional requirements for each group and single activity

Service	Monthly downtime	Response time level	File type	No of simul. users	Execution constraint
YMENS CRM	4 min	High	Office doc, PDF, audio, video	500	None
Zoho CRM	4 min	High	Office doc, PDF, audio, video	500	None
Sugar CRM	10 min	Medium	Office doc, PDF, audio, video	200	500 (monthly basis)
InvoiceNinja	4 min	High	Office doc, PDF, audio, video	600	None
Ninja Email	4 min	High	Office doc, PDF, audio, video	400	None
Simple Invoices	10 min	Medium	Office doc, PDF, audio, video	300	None
Mailjet	4 min	High	Office doc, PDF, audio, video	100	1K (monthly basis)
Open Source Billing	4 min	Medium	Office doc, PDF, audio, video	200	None
Gmail	4 min	High	Office doc, PDF, audio, video	100	None

First Main Step: Business Matchmaking. BM was used to identify the most suitable cloud services. As a first step, the questionnaire was applied on the whole BP (see starting notebook at Fig. 4).

We specified functional requirements in the first 3 questions - object *Send*, action *Invoice* and APQC category *9.2.2 Invoice Customer* - and none of the cloud services matched.

Next, the questionnaire was applied on two single activities (i.e., Manage Customer Relationship and Send Invoice) as well as on a group of activities (i.e., Create Invoice, Update Invoice and Check Invoice Completeness).

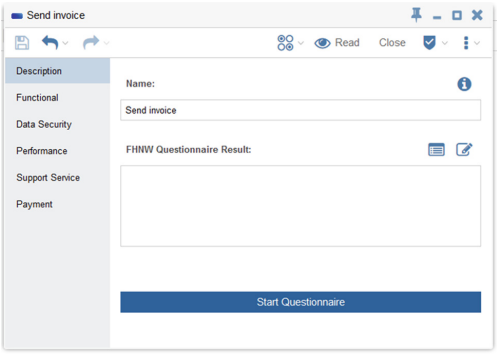


Fig. 4. The starting notebook for the whole process

Table 2 shows the functional requirements for each activity/group. In the first case, after specifying action, object and APQC category, the questionnaire showed the 3 matching cloud services: YMENS, Zoho and SugarCRM. In the 4th question, we chose the *Performance* category, and the question prioritisation algorithm kicked in. The question regarding the number of simultaneous users was asked (attribute with highest entropy) and a value of 500 was entered. This filtered out SugarCRM as it has the capability of max 200 simultaneous users.

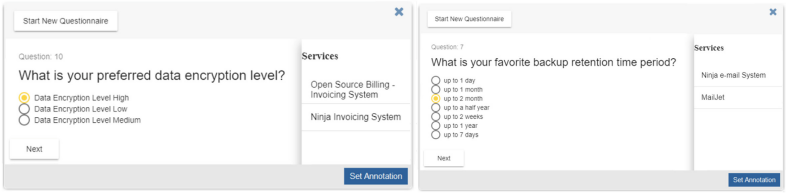
Table 2. Functional requirements for each group and single activity

BPMN activity	Action	Object	APQC category
Manage Customer Relationship *(Single activity)*	Manage	Customer Relationship	3.5.2.4 Manage Customer Relationship
Send Invoice *(Single activity)*	Send	Invoice	9.2.2.3 Transmit Billing Data to Customers
Create New Invoice, Update Invoice, Check Invoice Completeness *(Group of activities)*	Generate	Invoice	9.2.2.2 Generate Customer Billing Data

Similarly, we applied the questionnaire on the designated group of activities. The matching services were InvoiceNinja and Open Source Billing, see Fig. 5a.

Finally, we applied the questionnaire on the last BP activity: Send Invoice. The matching cloud services were Ninja E-mail and Mailjet (see Fig. 5b).

Second Main Step: Technical Matchmaking & Selection. As the final result maps to two services per each activity (group), we now proceed with the technical matching and selection. Suppose that the user provides the next global



(a) The selected invoice management services (b) The selected email services

Fig. 5. The selected services for last two activity groups

requirements for the whole process: $cost < 100$ euros per month, $cycletime < 1$ min and $VPM < 16$ (#vulnerabilities per month). Further, suppose that the user imposes for the *Manage Customer Relationship* activity the following constraints: $responsetime < 30$ s and $VPM < 10$. Finally, Table 3 depicts the non-functional profiles of the remaining services.

Table 3. The technical non-functional offerings of the 6 services

Service	Cost	Response time	VPM
ZOHO CRM	30 euros	35 s	10
YMENS CRM	35 euros	20 s	05
Mailjet	25 euros	10 s	02
Ninja_Email	10 euros	15 s	03
Open Source Billing	35 euros	25 s	08
InvoiceNinja	45 euros	10 s	05

Technical non-functional matching would then filter Zoho CRM as it does not conform to the local constraints posed for the CRM activity. This leads to selecting over 4 solutions as we have one candidate for the first (group) of activities and 2 candidates for the rest two activity groups. However, while running service selection, it is detected that the Ninja_Email and Open Source Billing are incompatible, which leaves us with 3 solutions. Moreover, the solution mapping to selecting YMENS, Open Source Billing and Ninja_Email has VPM equal to 16 violating the respective global constraint. So, in the end, we need to select between 2 solutions which are depicted in Table 4.

Table 4. The final ordered solutions produced

Solution	Cost	Cycle time	VPM	Utility
YMENS + InvoiceNinja + Ninja_Email	90 euros	45 s	13	0.144
YMENS + Open Source Billing + Mailjet	95 euros	50 s	15	0.099

As the broker requires to optimise all non-functional terms (cost, cycle time and VPM), it gives equal preference over them. By also considering that the activities are sequentially executed in the BPaaS workflow, the final result would map to selecting services YMENS, InvoiceNinja, and Ninja_Email. While there is perfect syntactic compatibility between InvoiceNinja and Ninja_Email as they are offered by the same company, in the case of YMENS CRM and InvoiceNinja the message types are compatible but still need to be aligned (e.g., attributes *accountid* and *id_number* mapping to the same attribute *id* of concept *Client*). As such, the MS service was included between these 2 services resulting in a workflow with 4 services sequentially executed (YMENS CRM → MS → InvoiceNinja → Ninja_Email).

7 Conclusions and Future Work

This paper has introduced a novel architecture for the design of BPaaS products able to effectively deal with the business-to-IT alignment problem in order to map an initial domain-specific BP into an executable BPaaS workflow. Such an architecture has been carefully designed and implemented to include suitable components which focus on different parts of the business-to-IT alignment problem, including business and technical matchmakers, a service selection as well as an automatic workflow update component to enable the effective addressing of the message compatibility problem in service-based workflow execution.

Our future work will focus on more advanced research challenges which include: (a) the automatic production of a more complete and more close to production workflow via the incorporation of different kinds of non-service tasks (see previous section); (b) the automatic population of the KB; (c) the coverage of additional cases in business-to-technical-requirement alignment.

Acknowledgments. This research has received funding from the European Community's Framework Programme for Research and Innovation HORIZON 2020 (ICT-07-2014) under grant agreement number 644690 (CloudSocket).

References

1. Antunes, G., Bakhshandeh, M., Borbinha, J., Cardoso, J., Dadashnia, S., Francescomarino, C.D., Dragoni, M., Fettke, P., Gal, A., Ghidini, C., Hake, P., Khiat, A., Klinkmüller, C., Kuss, E., Leopold, H., Loos, P., Meilicke, C., Niesen, T., Pesquita, C., Péus, T., Schoknecht, A., Sheetrit, E., Sonntag, A., Stuckenschmidt, H., Thaler, T., Weber, I., Weidlich, M.: The process model matching contest 2015. In: Lecture Notes in Informatics (2015)
2. Azzini, A., Braghin, C., Damiani, E., Zavatarelli, F.: Using Semantic Lifting for Improving Process Mining: A Data Loss Prevention System Case Study (2013)
3. Benaboud, R., Maamri, R., Sahnoun, Z.: Agents and owl-s based semantic web service discovery with user preference support. Int. J. Web Semant. Technol. **4**(2), 57–75 (2013)

4. Cliffe, O., Andreou, D.: Service Matchmaking Framework. Public Deliverable D5.2a, Alive EU Project Consortium, 10 September 2009. http://www.ist-alive.eu/index.php?option=com_docman&task=doc_download&gid=28&Itemid=49
5. Cloud Select Industry Group (C-SIG): Cloud Service Level Agreement Standardization Guidelines. Technical report, EC (2014)
6. Duipmans, E.: Business Process Management in the cloud: Business Process as a Service (BPaaS). Technical report (2012)
7. Emmenegger, S., Hinkelmann, K., Laurenzi, E., Martin, A., Thönssen, B., Witschel, H.F., Zhang, C.: An ontology-based and case-based reasoning supported workplace learning approach. In: Hammoudi, S., Pires, L.F., Selic, B., Desfray, P. (eds.) MODELSWARD 2016. CCIS, vol. 692, pp. 333–354. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66302-9_17
8. Emmenegger, S., Hinkelmann, K., Laurenzi, E., Thönssen, B.: Towards a procedure for assessing supply chain risks using semantic technologies. In: Fred, A., Dietz, J.L.G., Liu, K., Filipe, J. (eds.) IC3K 2012. CCIS, vol. 415, pp. 393–409. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-54105-6_26
9. Faria, D., Pesquita, C., Santos, E., Cruz, I.F., Couto, F.M.: AgreementMakerLight Results for OAEI 2013. http://disi.unitn.it/~p2p/OM-2013/oaei13_paper1.pdf
10. Frank, U.: Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering, pp. 133–157. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36654-3_6
11. Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., van der Merwe, A., Woitsch, R.: A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. Comput. Ind. **79**, 77–86 (2016)
12. Hinkelmann, K., Kritikos, K., Kurjakovic, S., Lammel, B., Woitsch, R.: A modelling environment for business process as a service. In: Krogstie, J., Mouratidis, H., Su, J. (eds.) CAiSE 2016. LNBIP, vol. 249, pp. 181–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39564-7_18
13. Hinkelmann, K., Laurenzi, E., Lammel, B., Kurjakovic, S., Woitsch, R.: A semantically-enhanced modelling environment for business process as a service. In: ES, pp. 143–152. IEEE (2016)
14. Hrgovcic, V., Karagiannis, D., Woitsch, R.: Conceptual modeling of the organisational aspects for distributed applications: the semantic lifting approach. In: COMPSAC Workshops, pp. 145–150. IEEE (2013)
15. Hwang, C., Yoon, K.: Multiple criteria decision making. In: Lecture Notes in Economics and Mathematical Systems (1981). https://doi.org/10.1007/978-3-642-48318-9
16. Jiang, S., Aagesen, F.A.: An approach to integrated semantic service discovery. In: Gaïti, D., Pujolle, G., Al-Shaer, E., Calvert, K., Dobson, S., Leduc, G., Martikainen, O. (eds.) AN 2006. LNCS, vol. 4195, pp. 159–171. Springer, Heidelberg (2006). https://doi.org/10.1007/11880905_14
17. Karagiannis, D., Buchmann, R.A., Burzynski, P., Reimer, U., Walch, M.: Fundamental conceptual modeling languages in OMiLAB. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 3–30. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_1
18. Klusch, M.: Semantic web service coordination. In: Schumacher M., Schuldt H., Helin H. (eds.) CASCOM: Intelligent Service Coordination in the Semantic Web, pp. 59–104 (2008). https://doi.org/10.1007/978-3-7643-8575-0_4

19. Kritikos, K., Magoutis, K., Plexousakis, D.: Towards knowledge-based assisted IaaS selection. In: CloudCom, pp. 431–439. IEEE Computer Society (2016)
20. Kritikos, K., Plexousakis, D.: Requirements for QoS-based web service description and discovery. IEEE Trans. Serv. Comput. **2**(4), 320–337 (2009)
21. Kritikos, K., Plexousakis, D.: Novel optimal and scalable nonfunctional service matchmaking techniques. IEEE Trans. Serv. Comput. **7**(4), 614–627 (2014)
22. Kritikos, K., Plexousakis, D.: Multi-cloud application design through cloud service composition. In: CLOUD, pp. 686–693. IEEE, New York (2015)
23. Kritikos, K., Plexousakis, D.: Towards combined functional and non-functional semantic service discovery. In: Aiello, M., Johnsen, E.B., Dustdar, S., Georgievski, I. (eds.) ESOCC 2016. LNCS, vol. 9846, pp. 102–117. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44482-6_7
24. Kritikos, K., Plexousakis, D., Paternò, F.: Task model-driven realization of interactive application functionality through services. TiiS **3**(4), 25 (2014)
25. Laurenzi, E., Hinkelmann, K., Reimer, U., van der Merwe, A., Sibold, P., Endl, R.: DSML4PTM - A domain-specific modelling language for patient transferal management. In: ICEIS 2017, Porto, Portugal, pp. 520–531 (2017)
26. Plebani, P., Pernici, B.: URBE: web service retrieval based on similarity evaluation. IEEE Trans. Knowl. Data Eng. **21**(11), 1629–1642 (2009)
27. Saati, T.: The Analytic Hierarchy Process. McGraw-Hill, New York (1980)
28. Silver, B.: BPMN Method and Style, 2nd edn. Cody-Cassidy Press, Aptos (2011)
29. Uschold, M., King, M., Morale, S., Zorgios, Y.: The enterprise ontology. Knowl. Eng. Rev. **13**(01), 31–89 (1998)
30. Vaishnavi, V., Kuechler, B.: Design Science Research in Information Systems (2004). http://desrist.org/desrist/content/design-science-research-in-information-systems.pdf
31. Watfa, M.K., Najjar, N.A.L., Cheikha, J., Buali, N.: A new framework for cloud business process management. In: Zhang, Y., Peng, L., Youn, C.-H. (eds.) CloudComp 2015. LNICST, vol. 167, pp. 83–92. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38904-2_9
32. Whibley, P.: BPM in the Cloud - Transforming the Business Case for Process Improvement. Technical report (2012)
33. Woitsch, R., Utz, W.: Business Processes as a Service (BPaaS): A Model-Based Approach to align Business with Cloud offerings (2015). https://zenodo.org/record/35583#.WbDscNhLfmE